PERANCANGAN DAN IMPLEMENTASI SISTEM KEAMANAN JARINGAN KOMPUTER DENGAN INTRUSION PREVENTION SYSTEM BERBASIS SNORT PADA JARINGAN DEMILITARIZED ZONE UNIVERSITAS NEGERI JAKARTA

DHANI WIDYA DARMA
5235107372

Skripsi Ini Dibuat Untuk Memenuhi Sebagian Persyaratan Dalam Memperoleh Gelar Sarjana Pendidikan

PROGRAM STUDI PENDIDIKAN TEKNIK INFORMATIKA DAN KOMPUTER
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS NEGERI JAKARTA
2015

PERANCANGAN DAN IMPLEMENTASI SISTEM KEAMANAN JARINGAN KOMPUTER DENGAN INTRUSION PREVENTION SYSTEM BERBASIS SNORT PADA JARINGAN DEMILITARIZED ZONE UNIVERSITAS NEGERI JAKARTA

DHANI WIDYA DARMA

ABSTRAK

Penelitian ini bertujuan untuk merancang dan mengimplementasi sebuah sistem keamanan jaringan dengan *Intrusion Prevention System* (IPS) yang dapat menghalau serangan dari dalam jaringan DMZ UNJ. Penelitian dilakukan di *data center* milik Pusat Teknologi Informasi dan Komunikasi (PUSTIKOM) UNJ pada bulan April sampai dengan Desember 2014. Metode yang digunakan pada penelitian ini adalah *Research and Development (R & D)*. Setelah melakukan perancangan sistem keamanan dengan *snort* yang dijadikan sebagai IPS, kemudian sistem keamanan jaringan tersebut diuji menggunakan teknik *DoS* dengan *Ping Flood* dan aplikasi *LOIC*. Dari data hasil uji coba yang dilakukan dengan mengirimkan paket *ping* dengan besaran lebih dari 32 *bytes* sebanyak 10 kali, sistem dapat melakukan pemblokiran terhadap paket tersebut begitu juga pengujian dengan aplikasi LOIC, dari total paket berbahaya sebanyak 12318 yang dikirimkan aplikasi LOIC, hanya 10 paket yang berhasil dikirim ke *server* tujuan dan sebanyak 12308 paket gagal dikirim. Berdasarkan pada hasil pengujian yang telah dilakukan, menunjukkan sistem keamanan yang dirancang dapat melakukan pemblokiran serangan yang dilakukan sehingga paket yang berupa serangan tidak diteruskan ke server tujuan.

Kata kunci: sistem keamanan jaringan, *intrusion prevention system*, *snort* dan *demilitarized zone*.

SYSTEM DESIGN AND IMPLEMENTATION OF NETWORK SECURITY USING INTRUSION PREVENTION SYSTEM BASED ON SNORT IN DEMILITARIZED ZONE NETWORK AT UNIVERSITAS NEGERI JAKARTA

DHANI WIDYA DARMA

ABSTRACT

This research aims to design and implement a network security system using Intrusion Prevention System (IPS) which is able to block attack from inside of DMZ network at UNJ. The research was conducted at the data center of Information and Communication Technology (PUSTIKOM) UNJ, starting from April to December 2014. This research using Research and Development (R&D) as methodology. After designing, configuring and modifying the snort that serves as IPS, the network security system was tested using the Ping Flood DoS technique and LOIC application. The data results shows by sending ping packets more than 32 bytes as many as 10 times, the system can perform the blocking of the packets, as well as testing the application LOIC, the total malicious packets transmitted as much as 12.318 LOIC applications, only 10 requests successfully delivered to the destination server and as many as 12.308 packets failed to be sent. Based on the results of tests showing that the security system can be designed to block the attacks so that the attack packets are not forwarded to the destination server.

Keyword: network security system, intrusion prevention system, snort and demilitarized zone.

HALAMAN PENGESAHAN

NAMA DOSEN	TANDA TANGAN	TANGGAL
M. Ficky Duskarnaen, S.T., M.Sc. (Dosen Pembimbing I)	.....	10/2 2015
Mochammad Djaohar, S.T., M.Sc. (Dosen Pembimbing II)	.....	10/2 2015

PENGESAHAN PANITIA UJIAN SKRIPSI

NAMA DOSEN	TANDA TANGAN	TANGGAL
Drs. Wisnu Djatmiko, M.T. (Ketua Penguji)	.....	13-02-2015
Prasetyo Wibowo Yunanto, M.Eng (Dosen Penguji)	.....	10-2-2015
Prof. Dr. Ir. Ivan Hanafi, M.Pd. (Dosen Ahli)	.....	10/2 2015

Tanggal Lulus: 27 Januari 2015

HALAMAN PERNYATAAN

Dengan ini saya menyatakan bahwa:

1. Karya tulis skripsi saya ini adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik sarjana, baik di Universitas Negeri Jakarta maupun di perguruan tinggi lain.
2. Karya tulis ini adalah murni gagasan, rumusan dan penelitian saya sendiri dengan arahan dosen pembimbing.
3. Dalam karya tulis ini tidak terdapat karya atau pendapat yang telah ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan dicantumkan dalam daftar pustaka.
4. Pernyataan ini saya buat dengan sesungguhnya dana apabila dikemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh karenan karya tulis ini, serta sanksi lainnya sesuai dengan norma yang berlaku di Universitas Negeri Jakarta.

Jakarta, 27 Januari 2015

Yang membuat pernyataan

Dhani Widya Darma

5235107372

KATA PENGANTAR

Segala puji dan syukur kehadirat Allah SWT. atas rahmat dan karuniaNya yang telah diberikan kepada peneliti, sehingga peneliti dapat menyelesaikan skripsi dengan judul: "Perancangan dan Implementasi Sistem Keamanan Jaringan Komputer dengan Intrusion Prevention System Berbasis Snort pada Jaringan Demilitarized Zone Universitas Negeri Jakarta".

Dalam merencanakan, menyusun dan menyelesaikan penulisan skripsi, peneliti banyak menerima bantuan, bimbingan dan motivasi serta dukungan dari berbagai pihak. Oleh karena itu, pada kesempatan ini peneliti bermaksud mengucapkan terima kasih yang disampaikan kepada:

1. Bapak Muhammad Ficky Duskarnaen, S.T., M.Sc. selaku pembimbing I dan Bapak Mochammad Djaohar, S.T., M.Sc. selaku pembimbing II yang telah memberikan motivasi, arahan dan kepercayaan kepada peneliti dalam menyelesaikan skripsi ini.
2. Orang tua yang tak hentinya memanjatkan doa dan memberikan semangat kepada peneliti dalam menyelesaikan skripsi ini.
3. PUSTIKOM Universitas Negeri Jakarta yang telah memberikan kesempatan kepada peneliti untuk melakukan penelitian di jaringan DMZ UNJ.

Peneliti menyadari bahwa skripsi ini jauh dari kesempurnaan, untuk itu peneliti mohon maaf apabila terdapat kekurangan dan kesalahan baik dari isi maupun penulisan. Akhir kata peneliti berharap agar penulisan dan penyusunan proposal penelitian ini dapat bermanfaat bagi para pembaca dan semua pihak yang terkait.

Jakarta, 27 Januari 2015

Dhani Widya Darma

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Keamanan merupakan bagian terpenting yang harus diperhatikan dalam menjaga suatu informasi. Seiring dengan perkembangan zaman, penyebaran informasi saat ini sangat dipengaruhi dengan pemanfaatan jaringan komputer untuk saling bertukar informasi yang membuatnya menjadi lebih mudah dan cepat. Jaringan komputer dapat diartikan sebagai kumpulan komputer yang saling terhubung (Madcoms, 2009:1). Hampir disetiap organisasi pemerintah ataupun swasta sudah memanfaatkan jaringan komputer untuk mempermudah arus pertukaran informasi yang terjadi di dalamnya. Akan tetapi dibalik kemudahannya itu, jaringan komputer tidak bisa lepas dari masalah keamanan yang dapat mempengaruhi kredibilitas dan ketersediaan suatu layanan informasi. Terlebih lagi pada jaringan komputer yang terhubung dengan Internet dimana banyak berbagai ancaman yang dapat berupa *virus, trojan, malware, cracker*, dan yang lainnya yang berpotensi untuk merusak sehingga membuat suatu jaringan komputer menjadi rentan terhadap penyusupan dan penyalahgunaan sistem yang dapat merugikan.

Dalam menjaga suatu jaringan komputer banyak cara yang dapat dilakukan mulai dari menggunakan *hardware* ataupun *software* baik yang bersifat *proprietary* atau *opensource.* Salah satu software keamanan yang dapat digunakan adalah *snort* yang merupakan *NIDS (Network-Based Intrusion Detection System)* (Arius, 2006:19). Sebagai *Intrusion Detection System* (IDS) yang bekerja pada sisi jaringan, *snort* bekerja dengan menggunakan prinsip *sniffer* yang berfungsi

mengawasi dan mendeteksi apabila terdapat paket data yang mencurigakan dan melaporkannya ke *administrator* jaringan untuk pengambilan tindakan lebih lanjut. Tetapi penggunaan snort sebagai IDS dirasa masih kurang efektif karena hanya dapat mendeteksi dan memberikan peringatan berupa *log* tanpa mampu mengambil tindakan lebih lanjut. Untuk itu perlu dikembangkan menjadi sebuah *Intrusion Prevention System (IPS)* yang memiliki semua kemampuan IDS dan juga dapat menghentikan insiden yang terjadi (Scarfone dan Mell, 2007:2-1) dengan melakukan beberapa modifikasi pada software IDS snort.

Universitas Negeri Jakarta (UNJ) memiliki jaringan komputer yang berpusat di PUSTIKOM. Secara garis besar, jaringan komputer yang dimiliki UNJ terdiri dari 3 jaringan utama yaitu jaringan internet (WAN), jaringan *Local Area Network* (LAN) dan *Demilitarized Zone* (DMZ). Jaringan WAN menghubungkan ke dua jaringan lainnya yaitu LAN dan DMZ dengan internet, kemudian jaringan LAN menghubungkan gedung-gedung yang ada di lingkungan UNJ, sedangkan jaringan DMZ merupakan jaringan tempat diletakannya *server-server* yang dimiliki oleh UNJ.

Saat ini UNJ menggunakan sistem keamaanan jaringan berupa *firewall* yang dipasang untuk melindungi jaringan LAN dan DMZ dari ancaman yang berasal dari luar jaringan UNJ. Akan tetapi penggunaan *firewall* tersebut masih belum maksimal, hal ini didukung dengan adanya laporan dari *firewall* pada tanggal 28 April 2014 terjadi aktivitas tidak wajar pada jaringan *DMZ* dimana terdapat *server* yang melakukan *generate traffic* dengan prinsip *DoS* dengan target alamat IP 39.50.197.165 yang dimiliki oleh perusahan telekomunikasi yang berada di Pakistan sebanyak 1769 insiden. Selain itu, UNJ pernah mendapat surat

pemberitahuan bahwa *server* yang dimiliki UNJ melakukan serangan ke *server* milik pihak ketiga yaitu NETpilot GmbH yang berada di Internet. Dengan kata lain *firewall* yang saat ini digunakan hanya untuk menghalau serangan yang berasal dari luar jaringan DMZ UNJ tetapi tidak untuk menghalau serangan yang berasal dari dalam jaringan DMZ UNJ. Keterbatasan kemampuan *administrator* dalam menanggapi suatu serangan juga berpengaruh terhadap ketersediaan layanan informasi yang dimiliki oleh UNJ.

Oleh karena itu, dibutuhkan perancangan suatu sistem yang dapat melakukan pendeteksian dan pencegahan terhadap ancaman yang berasal dari dalam jaringan komputer yang ada di Universitas Negeri Jakarta khususnya jaringan DMZ secara cepat dan otomatis dalam membantu *administrator* menjaga keamanan dan ketersediaan layanan informasi. Hal tersebut melatarbelakangi dibuatnya sebuah sistem keamanan jaringan komputer sebagai pendeteksi dan pencegahan ancaman yang berbasis s*nort*.

1.2 Identifikasi Masalah

Berdasarkan latar belakang di atas, masalah yang dapat diidentifikasi sebagai berikut:

1. Saat ini sistem keamanan milik UNJ yang berupa *firewall* hanya dapat menghalau serangan yang berasal dari luar jaringan DMZ Universitas Negeri Jakarta.
2. Terdapat server pada jaringan DMZ UNJ yang melakukan *generate traffic* dengan *target* alamat IP 39.50.197.165 menggunakan prinsip *DoS*.

3. Adanya pengaduan dari pihak ketiga yaitu NETpilot GmbH yang menyatakan bahwa *server* milik UNJ melakukan serangan terhadap *server* miliknya yang terhubung di Internet.
4. Dalam penanganannya masih bergantung kepada kecepatan, ketersedian dan kemampuan administrator dalam menanggapi suatu serangan.

1.3 Pembatasan Masalah

Untuk menghindari pembahasan yang meluas, maka pembahasan masalah dibatasi pada hal-hal berikut:

1. Pengujian dilakukan di Data Center PUSTIKOM UNJ.
2. Sistem keamanan diletakkan antara *firewall* dan jaringan DMZ.
3. Jaringan yang dimaksud merupakan milik UNJ.
4. Sistem keamanan yang dirancang dapat mendeteksi dan melakukan pemblokiran serangan.
5. Serangan yang dilakukan berupa serangan *Denial of Service (DoS)* berupa *Ping Flood* dan *generate traffic* dengan aplikasi LOIC.
6. Tools keamanan yang digunakan dalam perancangan sistem keamanan ini adalah *snort* serta beberapa tools pendukung lainnya.

1.4 Perumusan Masalah

Berdasarkan latar belakang, identifikasi, dan pembatasan masalah, maka perumusan masalah yang akan dibahas pada penelitian ini adalah:

Apakah penggunaan sistem keamanan jaringan komputer dengan intrusion prevention system berbasis snort dapat menghalau serangan yang berasal dari dalam jaringan DMZ Universitas Negeri Jakarta?

1.5 Tujuan Penelitian

Tujuan umum penelitian ini adalah merancang sistem keamanan jaringan komputer dengan *Intrusion Prevention System (IPS)* berbasis *snort* untuk menjaga keamanan jaringan dan ketersedian layanan informasi di Universitas Negeri Jakarta.

1.6 Kegunaan Penelitian

Kegunaan dari penelitian ini diharapkan dapat membantu *administrator* jaringan dalam upaya menjaga ketersediaan layanan informasi di Universitas Negeri Jakarta, antara lain:

a. Menjaga keamanan jaringan UNJ.
b. Menghalau serangan yang berasal dari jaringan UNJ.
c. Memberikan laporan berupa log event serangan yang terjadi.

BAB II
KAJIAN TEORETIS DAN KERANGKA BERPIKIR

2.1 Kajian Teoretis

2.1.1 Definisi Jaringan Komputer

Jaringan komputer merupakan sekumpulan komputer berjumlah banyak yang terpisah-pisah akan tetapi saling berhubungan dalam melaksanakan tugasnya (Tanenbaum, 2003:2).

Tujuan dari jaringan komputer ini adalah agar setiap *komputer* dapat saling berbagi sumber daya, akses informasi, dan dapat saling berkomunikasi antara *node* yang satu dengan yang lainnya.

2.1.1.1 Jenis-Jenis Jaringan Komputer

Berdasarkan jangkauan area atau lokasi, jaringan komputer dibedakan menjadi 3 jenis yaitu:

1. Lo*cal Area Network (LAN)*

   Local Area Network (LAN) merupakan jaringan yang menghubungkan sejumlah komputer yang ada dalam suatu lokasi dengan area terbatas seperti ruang atau gedung pada sebuah sekolah maupun area gedung perkantoran. Biasanya pada jaringan ini setiap komputer/node dapat mengakses data dari komputer lain, menggunakan perangkat/periferal lain yang terhubung dengan jaringan seperti printer. Jumlah komputer/node yang terhubung pada LAN relatif kecil dan kebanyakan menggunakan kabel sebagai media penghubung.

2. *Metropolitan Area Network (MAN)*

   Metropolitan Area Network (MAN) merupakan jenis jaringan yang lebih besar dari LAN. Sebuah MAN terdiri dari beberapa jaringan LAN yang saling

terhubung dalam lingkup area yang lebih luas seperti suatu wilayah pada satu provinsi. Sebagai contoh, jaringan bank dimana beberapa kantor cabang sebuah bank di dalam sebuah kota besar dihubungkan antara satu dengan lainnya.

3. *Wide Area Network (WAN)*

Wide Area Network (WAN) merupakan jenis jaringan yang memberikan layanan lebih luas lagi dibanding MAN yaitu dapat menghubungkan suatu negara bahkan benua. WAN biasanya menggunakan satelit dan kabel bawah laut untuk berhubungan satu sama lain.

Selain itu terdapat 2 tipe jaringan yang dapat digunakan dalam mengatur sebuah jaringan komputer, antara lain (Madcoms, 2009:2):

1. *Client-Server*

Tipe jaringan *client-server* menghubungkan komputer *server* dengan beberapa komputer *client/workstation*. Komputer *server* adalah komputer yang menyediakan fasilitas atau layanan bagi komputer-komputer lain yang terhubung dalam jaringan. Sedangkan komputer *client* adalah komputer-komputer yang menggunakan fasilitas atau layanan yang diberikan oleh komputer *server*. Biasanya komputer *server* pada sebuah jaringan disebut juga dengan *Dedicated Server* karena komputer yang digunakan hanya sebagai penyedia fasilitas atau layanan untuk komputer *client/workstation*.

2. *Peer-to-Peer*

Tipe jaringan peer-to-peer menghubungkan beberapa komputer dalam sebuah jaringan. Pertukaran data dapat dilakukan antar komputer yang

terhubung tanpa perantara komputer server. Masing-masing komputer dapat berperan sebagai komputer server sekaligus sebagai komputer client.

2.1.1.2 Protokol Jaringan

Protokol merupakan perjanjian antara para pihak yang berkomunikasi yang mengatur bagaimana komunikasi itu di proses (Tanenbaum, 2003:27). Aturan tersebut meliputi tata cara atau metode yang mengatur bagaimana mengakses sebuah jaringan, topologi fisik, media penghubung yang digunakan, serta kecepatan transfer.

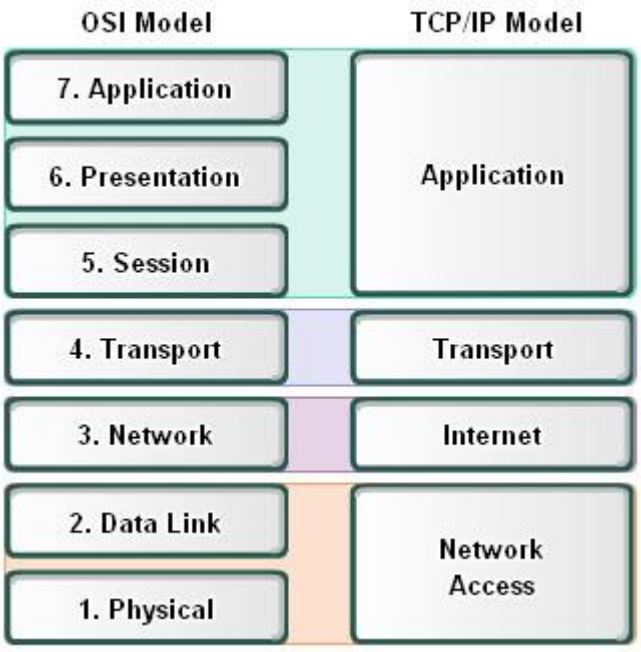
Pada awalnya sistem jaringan komputer sangat bergantung pada setiap vendor dalam penggunaan protokol, yang berarti setiap vendor memiliki protokol sendiri yang hanya berlaku untuk perangkat yang dibuatnya begitu juga vendor yang lain. Pada sebuah jaringan yang besar terdapat banyak protokol jaringan yang berbeda, karena tidak adanya satu protokol yang sama sehingga membuat banyak perangkat tidak bisa saling komunikasi. Untuk dapat membuat perangkat bisa saling berkomunikasi maka diperlukan suatu aturan yang baku dan disetujui oleh semua pihak. Oleh karena itu pada tahun 1977 sebuah organisasi di eropa yang bernama *International Organization for Standardization (ISO)* mengembangkan sebuah artsitektur jaringan yang mengatur bagaimana sebuah perangkat dapat saling berkomunikasi yang disebut model referensi *OSI (Open Systems Interconnection).* Model referensi OSI ini terdiri dari 7 lapisan, yang mengatur dari lapisan fisik hingga pada lapisan aplikasi yang dapat dilihat pada tabel berikut:

Tabel 2.1 Lapisan OSI

Lapisan	Nama Lapisan	Fungsi	Protokol	Keterangan
7	Aplikasi	Menyediakan pelayanan yang mendukung aplikasi pengguna.	Transaksi File (File Transfer), email, akses ke pangkalan data (database access)	Application Layer Protocol
6	Penyampaian	Menterjemahkan, kompresi, dan enkripsi data.	ASCII, EBCDIC, MIDI, MPEG, TIFF, JPEG, PICT, Quicktime	
5	Sesi	Mengkoordinasi komunikasi diantara sistem	SQL, NETBEUI, RPC, XWindows	
4	Transport	Memungkinkan paket data dikirim tanpa kesalahan dan tanpa duplikat.	TCP, UDP, SPX	Communication Layer Protocol
3	Jaringan	Menentukan jalur pengiriman dan melanjutkan paket ke alamat peralatan lain yang berjauhan. Pada lapisan ini data dikirim dalam bentuk paket.	IP, IPX, ARP, RARP, ICMP, RIP, OSFT, BOP	
2	Link Data	Mengatur data binari (0 dan 1) menjadi kumpulan logikal (logical group).	SUP, PPP, MTU	Physical Topology
1	Fisik	Penghantaran data binari melalui jalur komunikasi	10BaseT, 100BaseTX, 1000BaseTX, HSSI, V3.5, X2.1	

Model referensi ini pada awalnya ditujukan sebagai standard untuk mengembangkan protokol-protokol jaringan, sehingga dapat menggantikan implementasi protokol yang dimiliki oleh masing-masing vendor perangkat telekomunikasi. Meskipun banyak protokol yang berguna telah dikembangkan

dalam konteks OSI, model referensi dengan 7 lapis secara keseluruhan belum berkembang. Akan tetapi protokol TCP/IP yang saat ini lebih banyak digunakan dengan basis pada model referensi DARPA menjadikannya lebih populer. Protokol TCP/IP ini adalah sekelompok protokol yang mengatur komunikasi data dalam proses tukar-menukar data dari satu komputer ke komputer lain di dalam jaringan internet yang akan memastikan pengiriman data sampai ke alamat yang dituju. Protokol ini tidaklah dapat berdiri sendiri, karena memang protokol ini berupa kumpulan protokol *(protocol suite).* Berikut perbandingan antara protokol model referensi OSI dan TCP/IP:



Gambar 2.1 Perbandingan Model Referensi OSI dan TCP/IP

Protokol TCP/IP dikembangkan pada akhir dekade 1970-an hingga awal 1980-an sebagai sebuah protokol standar untuk menghubungkan komputer-komputer dan jaringan untuk membentuk sebuah jaringan yang luas (WAN). Sebagai sebuah standar jaringan terbuka, TCP/IP bersifat independen terhadap mekanisme transport jaringan fisik yang digunakan, sehingga dapat digunakan di mana saja.

Protokol ini menggunakan skema pengalamatan yang sederhana yang disebut sebagai alamat IP *(IP Address)* yang mengizinkan hingga beberapa ratus juta komputer untuk dapat saling berhubungan satu sama lainnya di Internet. Protokol ini juga bersifat *routable* yang berarti protokol ini cocok untuk menghubungkan sistem-sistem berbeda (seperti Microsoft Windows dan keluarga UNIX) untuk membentuk jaringan yang heterogen. Karena pertumbuhan internet dan protokol TCP/IP menjadikan model referensi OSI kurang berkembang dengan 7 lapisan yang membuatnya terlalu kompleks, jika dibandingkan dengan protokol TCP/IP yang memiliki lapisan yang lebih sedikit. Berikut penjelasan dari masing-masing lapisan yang terdapat pada protokol TCP/IP:

a. Lapisan Network Access

   Lapisan ini sama dengan lapisan Data Link dan Fisik pada model OSI. Lapisan ini mengatur pengalamatan perangkat keras dan pengiriman data fisik (Cloara, dkk., 2008:33). Adapun protokol yang termasuk pada lapisan ini antara lain *Ethernet, Fast Ethernet, Token Ring,* dan FDDI (*Fiber Distributed Data Interface*) (Cloara, dkk., 2008:27).

b. Lapisan Internet

   Lapisan internet sama dengan lapisan Network pada model OSI. Lapisan Internet memiliki fungsi sebagai penyedia fungsi IP addressing, routing, dan menentukan path terbaik. Protokol yang berhubungan dalam pengiriman paket meliputi IP, ICMP, ARP, RARP, dan Proxy ARP (Cloara, dkk., 2008:31).

c. Lapisan Transport

Lapisan Transport sama dengan lapisan Transport pada model OSI. Lapisan transport juga dikenal sebagai lapisan Host-to-Host. Tidak hanya bertanggung jawab untuk pengiriman data yang dapat diandalkan, tetapi lapisan ini juga dapat memastikan bahwa data tiba dalam urutan yang tepat (Cloara, dkk., 2008:27). Terdapat dua protokol yang termasuk dalam lapisan ini yaitu TCP dan UDP.

d. Lapisan Application

Lapisan ini merupakan gabungan fungsi dari ketiga lapisan atas pada model OSI yaitu *session, presentation, dan application*. Beberapa aplikasi populer menggunakan antarmuka lapisan ini untuk berkomunikasi dengan aplikasi pada jaringan (Cloara, dkk., 2008:26). Berikut protokol yang termasuk ke dalam lapisan ini, antara lain Telnet, HTTP/HTTPS, FTP, TFTP, DNS, SMTP, POP3, NFS, NNTP, SNMP, NTP, DHCP, dan sebagainya (Cloara, dkk., 2008:27).

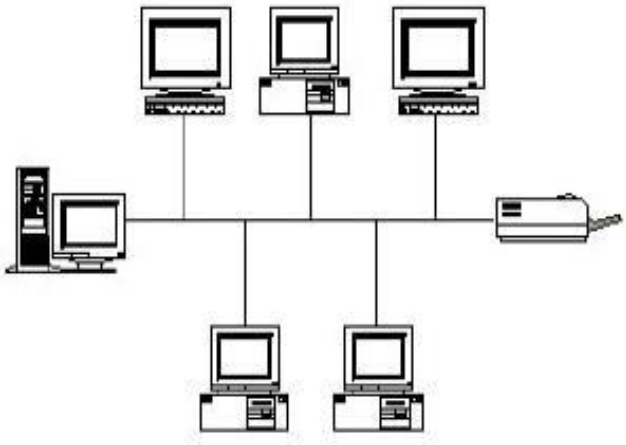
2.1.1.3 Topologi Jaringan

Topologi jaringan adalah suatu cara yang digunakan untuk menghubungkan antara komponen-komponen jaringan yang meliputi komputer server, komputer client, hub/switch, dan komponen jaringan lainnya (Madcoms, 2009:4). Dimana penggunaan topologi jaringan didasarkan pada biaya, kecepatan akses data, ukuran maupun tingkat konektivitas yang akan mempengaruhi kualitas maupun efiensi suatu jaringan.

Terdapat beberapa topologi jaringan yang dapat digunakan sesuai dengan kondisi yang ada dilapangan, antara lain:

1. Topologi Bus

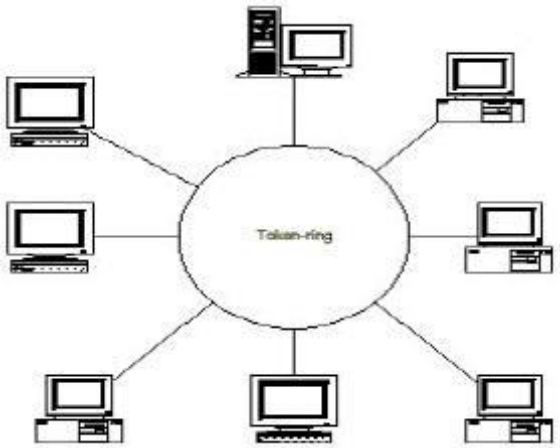
Topologi bus merupakan topologi yang menghubungkan beberapa komputer menggunakan kabel tunggal dimana setiap komputer dihubungkan secara serial disepanjang kabel. Karena hanya menggunakan kabel tunggal untuk berkomunikasi maka setiap perangkat harus bergantian dalam menggunakan jalur yang ada.



Gambar 2.2 Topologi Bus

2. Topologi Ring

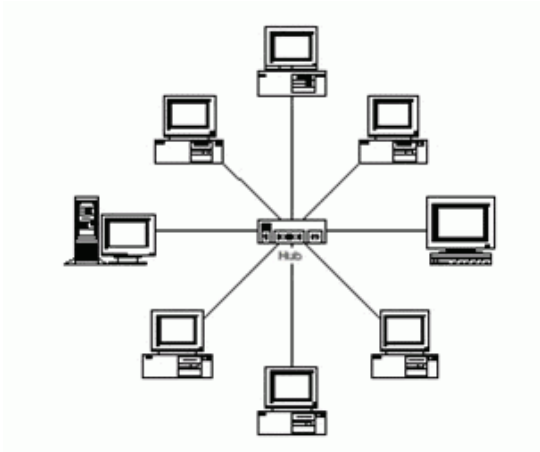
Topologi ring menghubungkan seluruh komputer menjadi satu bentuk lingkaran (ring) yang tertutup. Setiap komputer akan menerima dan melewatkan informasi dari satu komputer ke komputer yang lainnya, bila alamat-alamat yang di maksud sesuai maka informasi diterima dan bila tidak informasi akan di lewatkan.



Gambar 2.3 Topologi Ring

3. Topologi Star

Topologi star merupakan topologi yang menghubungkan beberapa komputer secara langsung ke hub/switch dimana perangkat ini sebagai pengontrol/pengatur lalu lintas seluruh komputer yang terhubung dalam jaringan.

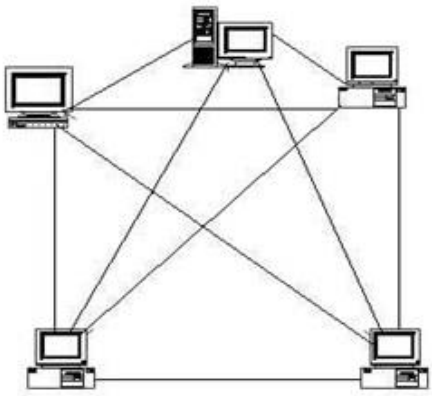


Gambar 2.4 Topologi Star

4. Topologi Mesh

Topologi mesh merupakan rangkaian jaringan yang saling terhubung secara mutlak dimana setiap perangkat komputer akan terhubung secara langsung ke setiap titik perangkat lainnya. Setiap komputer akan mempunyai

titik yang siap untuk berkomunikasi secara langsung dengan titik perangkat komputer lain yang menjadi tujuannya.

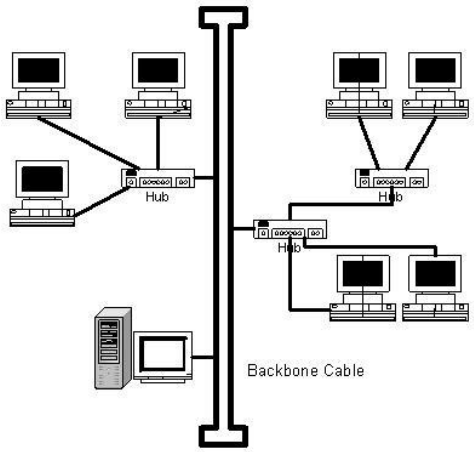


Gambar 2.5 Topologi Mesh

5. Topologi Tree

Topologi tree ini merupakan hasil pengembangan dari topologi star dan topologi bus yang terdiri dari kumpulan topologi star dan dihubungkan dengan satu topologi bus. Topologi tree biasanya disebut juga topologi jaringan bertingkat dan digunakan interkoneksi antar sentral.

Pada jaringan ini memiliki beberapa tingkatan simpul yang ditetapkan dengan suatu hirarki, gambarannya adalah semakin tinggi kedudukannya maka semakin tinggi pula hirarkinya. Setiap simpul yang memiliki kedudukan tinggi dapat mengatur simpul yang memiliki kedudukan yang rendah. Data dikirim dari pusat simpul kemudian bergerak menuju simpul rendah dan menuju ke simpul yang lebih tinggi terlebih dahulu.



Gambar 2.6 Topologi Tree

2.1.1.4 Perangkat Jaringan

2.1.1.4.1 Network Interface

Network interface merupakan perangkat yang digunakan untuk menghubungkan komputer dengan komputer lain pada sebuah jaringan agar dapat saling berkomunikasi. *Network interface* dapat menggunakan kabel coaxial, *twisted pair*, dan juga *wireless* sebagai media transmisinya (Madcoms, 2009:9). Gambar 2.7 menunjukan salah satu contoh dari network interface.

Gambar 2.7 Network Interface

2.1.1.4.2 Hub

Hub merupakan perangkat jaringan yang bekerja di OSI layer 1 *(physical)*. Hub berfungsi sebagai penerima sinyal dari sebuah komputer, kemudian mentransmisikan ke komputer lain yang terhubung pada jaringan. Hub tidak mengenal MAC Address, sehingga tidak dapat memilah data yang harus ditransmisikan dan yang tidak, hal ini dapat menyebabkan terjadinya *collision* pada sebuah jaringan (Madcoms, 2009:10). Gambar 2.8 menunjukan salah satu bentuk hub.

Gambar 2.8 Hub

2.1.1.4.3 Switch

Switch merupakan perangkat jaringan yang bekerja pada OSI Layer 2 *(data link)*. Switch berfungsi hampir sama dengan hub, akan tetapi switch mengenal *MAC Address* digunakan untuk memilah data mana yang harus ditransmisikan. Switch menampung daftar *MAC Address* perangkat yang dihubungkan melalui port-port yang digunakan untuk menentukan kemana harus mengirim paket, sehingga akan mengurangi *traffic* dan terjadinya *collision* pada jaringan (Madcoms, 2009:11). Switch ditunjukan pada Gambar 2.9.

Gambar 2.9 Switch

2.1.1.4.4 Router

Router (seperti gambar 2.10) merupakan perangkat jaringan yang bekerja pada OSI Layer 3 (Network). Router berfungsi sebagai penghubung/penerus paket data antara dua segmen jaringan atau lebih (Madcoms, 2009:11).

Gambar 2.10 Router

2.1.1.4.5 Repeater

Repeater merupakan perangkat yang digunakan untuk menguatkan sinyal. Repeater digunakan apabila menghubungkan perangkat dengan jarak yang berjauhan. Pada network interface, kualitas transmisi data hanya dapat bertahan dalam range waktu tertentu dan dalam jangkauan yang terbatas yang selanjutnya akan mengalami degradasi. Repeater akan berusaha mempertahankan integritas

sinyal dan mencegah dgradasi sampai paket-paket data terkirim sampai di tujuan. Gambar 2.11 menunjukan salah satu contoh repeater.

Gambar 2.11 Repeater

2.1.2 Jaringan Demilitarized Zone (DMZ)

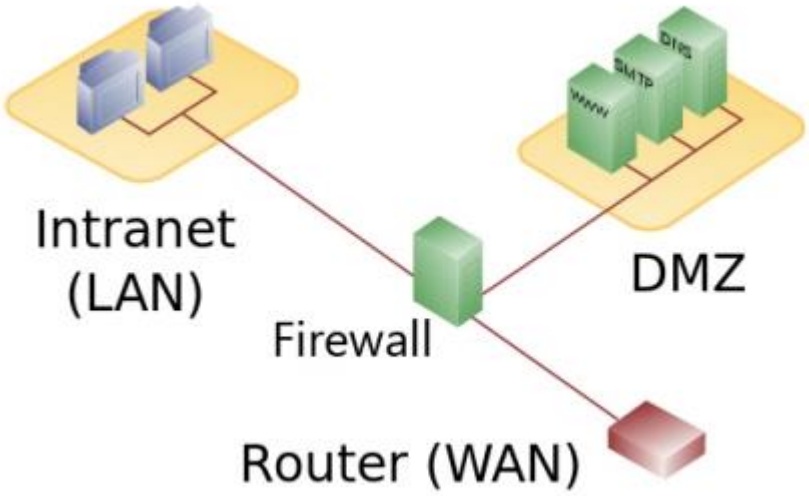
Demilitarized Zone (DMZ) merupakan area netral diantara jaringan publik (internet) dan jaringan intranet perusahaan yang dilindungi oleh firewall yang membatasi akses dari jaringan luar menuju host yang terletak di jaringan LAN (Lammle dan Timm, 2003:368). Tujuan dari adanya DMZ adalah sebagai pembatas jaringan yang dapat diakses dari luar (Internet), sehingga apabila terjadi aksi serangan atau penyusupan yang terganggu hanya jaringan DMZ saja dan tidak sampai pada jaringan internal dari organisasi atau perusahaan.

Ada beberapa cara yang dapat digunakan untuk merancang sebuah jaringan dengan DMZ. Dua metode paling mendasar adalah dengan *firewall* tunggal dan dengan *firewall* ganda.

2.1.2.1 DMZ dengan *Firewall* Tunggal

Sebuah *firewall* tunggal sedikitnya membutuhkan tiga antarmuka jaringan yang dapat digunakan untuk membuat sebuah arsitektur jaringan yang berisi

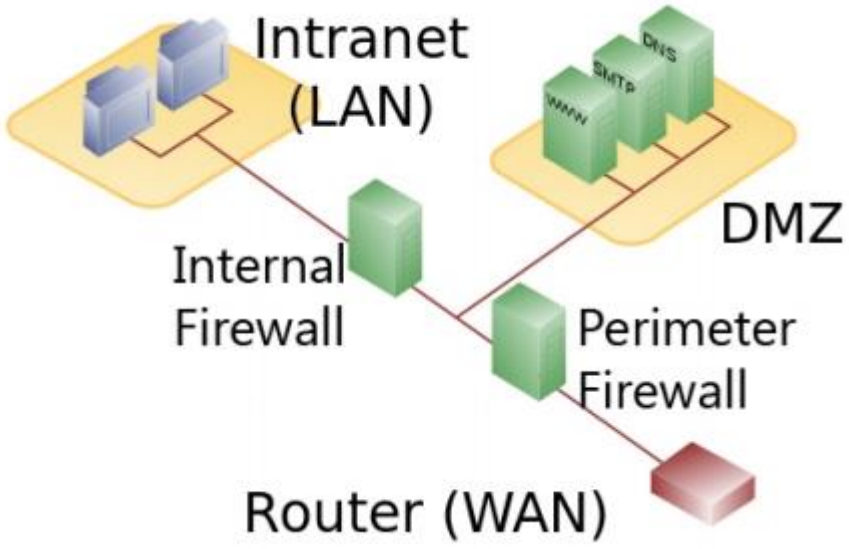
DMZ. Eksternal jaringan terbentuk dari ISP ke firewall pada antarmuka jaringan pertama, jaringan internal yang dibentuk dari network interface kedua, dan DMZ terbentuk dari antarmuka jaringan ketiga. Dengan metode ini, firewall menjadi titik pusat dari jaringan. Artinya firewall harus mampu menangani semua lalu lintas data dari dan ke jaringan internal serta DMZ (Kurniadi, 2010). Gambar 2.12 mengilustrasikan DMZ dengan firewall tunggal.



Gambar 2.12 DMZ dengan *Firewall* Tunggal

2.1.2.2 DMZ dengan *Firewall* Ganda

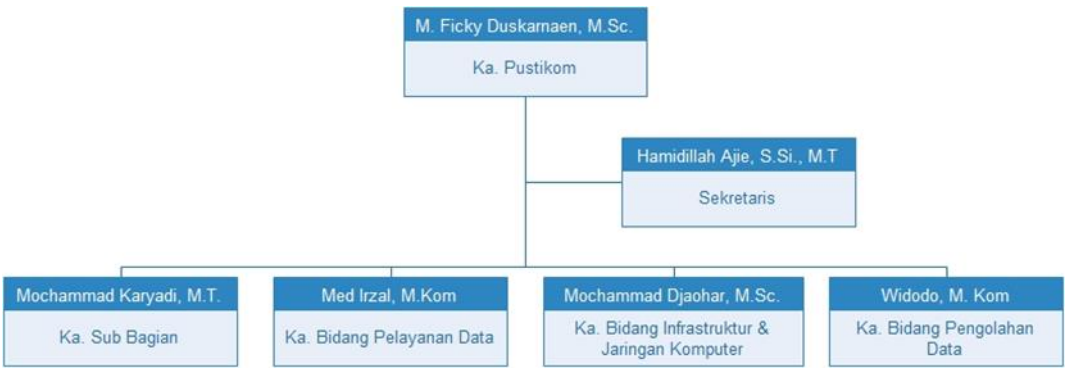
Pendekatan yang lebih aman adalah dengan menggunakan dua *firewall* untuk membuat DMZ. *Firewall* pertama disebut dengan *front-end firewall* atau perimeter yang dikonfigurasi hanya mengizinkan lalu lintas data menuju DMZ, sedangkan firwall kedua atau disebut juga dengan *back-end firewall* dikonfigurasi hanya untuk mengizinkan lalu lintas dari DMZ ke jaringan internal atau LAN (Kurniadi, 2010). Ilustrasi DMZ dengan *firewall* ganda dapat dilihat pada gambar 2.13.



Gambar 2.13 DMZ dengan *Firewall* Ganda

2.1.3 Jaringan Komputer Universitas Negeri Jakarta

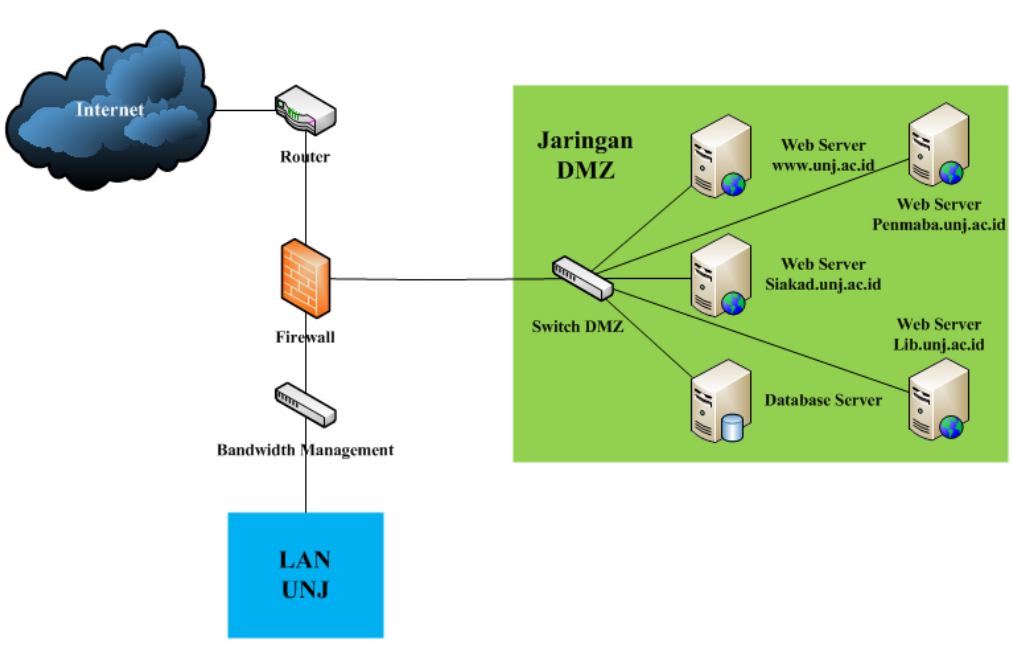
Pusat Teknologi Informasi dan Komunikasi (PUSTIKOM) merupakan salah satu unit pelaksana yang mengelola pemanfaatan teknologi informasi dan komunikasi untuk mendukung kegiatan pendidikan, penelitian, pengabdian kepada masyarakat, penyimpanan dan pengolahan data di Universitas Negeri Jakarta. Tercapainya peningkatan kuantitas dan kualitas layanan bidang teknologi informasi di lingkungan UNJ merupakan tujuan dari PUSTIKOM. PUSTIKOM mempunyai struktur organisasi yang digambarkan pada Gambar 2.14.



Gambar 2.14 Struktur Organisasi PUSTIKOM

Dari struktur ogranisasi tersebut, masing-masing bagian mempunyai tugas fungsionalitas untuk mewujudkan suatu tujuan. Jaringan komputer merupakan salah satu layanan yang dimiliki oleh PUSTIKOM untuk meningkatkan dan mendukung kegiatan yang ada di UNJ. Bagian jaringan komputer dan infrastruktur *data center* yang berada di PUSTIKOM merupakan tanggung jawab dari Kepala Bidang Infrastruktur dan Jaringan Komputer.

Secara umum jaringan komputer yang dimiliki UNJ terdiri dari 3 jaringan utama yaitu jaringan internet (WAN), jaringan *Local Area Network* (LAN) dan *Demilitarized Zone* (DMZ). Jaringan WAN menghubungkan ke dua jaringan yaitu LAN dan DMZ dengan internet, kemudian jaringan LAN menghubungkan gedung-gedung yang ada dilingkungan UNJ, sedangkan jaringan DMZ merupakan jaringan tempat diletakannya server-server yang dimiliki oleh UNJ. Untuk lebih jelasnya dapat dilihat pada Gambar 2.15.



Gambar 2.15 Jaringan Komputer UNJ

2.1.4 Keamanan Jaringan Komputer

Keamanan jaringan komputer merupakan proses pencegahan yang dilakukan oleh penyerang untuk terhubung ke dalam jaringan komputer melalui akses yang tidak sah, atau penggunaan secara ilegal dari komputer dan jaringan (Dewannanta, 2012). Pertumbuhan Internet yang sampai saat ini mengalami peningkatan berperan sangat besar terhadap penyebaran informasi. Semakin hari informasi yang berada di Internet semakin banyak, akurat, dan beberapa merupakan suatu informasi penting dan berharga sehingga perlu adanya perlakuan yang lebih spesifik terhadap informasi tersebut apalagi yang menyangkut privasi seseorang. Dengan semakin penting dan berharganya sebuah informasi tersebut, tentu menarik perhatian beberapa pihak yang tidak bertanggung jawab untuk memanfaatkan informasi tersebut demi keuntungan pribadi.

Keamanan jaringan memiliki beberapa tujuan yang berkaitan dengan suatu informasi, antara lain:

1. *Confidentiality*, merupakan usaha untuk menjaga informasi dari pengguna yang tidak berhak mengakses. *Confidentiality* biasanya berhubungan dengan informasi yang diberikan ke pihak lain.
2. *Integrity*, keaslian pesan yang dikirim melalui sebuah jaringan dan dapat dipastikan bahwa informasi yang dikirim tidak dimodifikasi oleh pengguna yang tidak berhak dalam perjalanan informasi tersebut.
3. *Availability*, aspek *availability* atau ketersediaan berhubungan dengan ketersediaan akses terhadap informasi yang dibutuhkan. Sistem informasi yang diserang dapat menghambat atau meniadakan akses ke informasi.

2.1.5 Jenis-Jenis Serangan Jaringan Komputer

Banyak cara yang dapat dilakukan seorang penyusup dalam mencapai suatu tujuan, dari ingin memperoleh sebuah data hingga ingin melumpuhkan sebuah sistem. Berikut ini adalah jenis-jenis serangan yang biasanya terjadi pada jaringan komputer:

1. *Denial of Services*

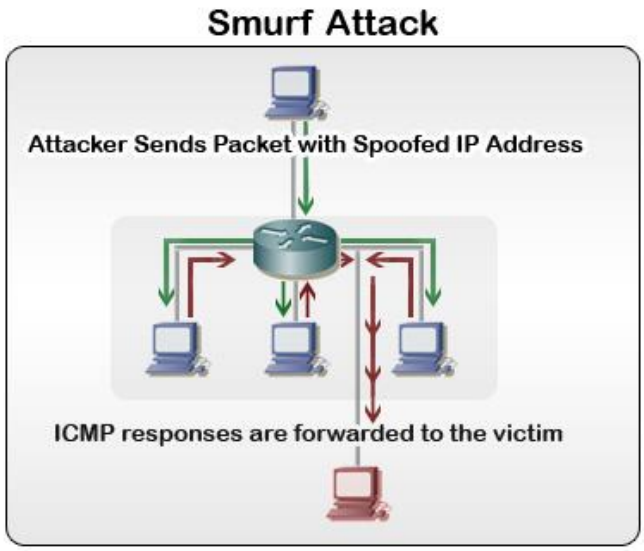
   Denial of Services (DoS) merupakan suatu upaya serangan dengan cara menurunkan kinerja suatu sistem secara terus menerus dengan cara mengulangi permintaan *(request)* ke *server* dari beberapa sumber secara simultan (Arius, 2006:53). DoS juga merupakan serangan yang dilancarkan melalui paket-paket tertentu, biasanya paket-paket sederhana dengan jumlah yang sangat besar dan banyak bermaksud mengacaukan keadaan jaringan korbannya. Beberapa metode yang dapat digunakan pada serangan DoS seperti:

   a. *Smurf Attack*

      Serangan jenis ini biasanya dilakukan dengan menggunakan *IP spoofing*, yaitu mengubah alamat IP dari datangnya *request*. Penggunaan *IP spoofing* memungkinkan respon dari *ping* tadi dialamatkan ke komputer yang IP-nya di *spoof*. Akibatnya, komputer tersebut akan menerima banyak paket. Hal tersebut dapat mengakibatkan pemborosan penggunaan bandwidth jaringan yang menghubungkan komputer tersebut (Arius, 2006:49).

      Seseorang yang melakukan *smurf attack* biasanya membanjiri *router* dengan paket permintaan *Internet Control Message Protocol (ICMP)*.

Oleh karena alamat IP tujuan pada paket yang dikirim adalah alamat *broadcast* dari jaringan, maka *router* akan mengirimkan *ICMP* ke semua host yang ada pada jaringan. Apabila ada banyak *host* dijaringan, maka akan terjadi trafik *ICMP response* dan permintaan dalam jumlah yang sangat besar seperti pada ilustrasi gambar 2.16 berikut:

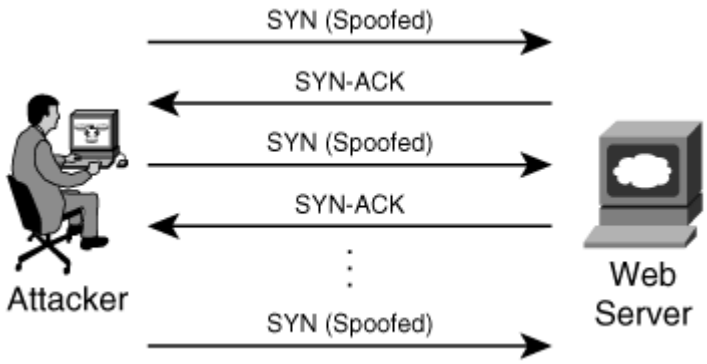


Gambar 2.16 Ilustrasi *Smurf Attack*

b. *SYN Attack*

S*YN attack* atau *SYN flood* adalah serangan yang dilakukan apabila ditemukan kelemahan dari spesifikasi TCP/IP. Paket SYN dikirimkan pada saat memulainya *handshake* (terkoneksi) antara aplikasi sebelum pengiriman data dilakukan (Arius, 2006:52). Pada kondisi normal, aplikasi klien akan mengirimkan paket TCP SYN untuk mensinkronisasi paket pada aplikasi di server (penerima). Server akan mengirimkan respon berupa acknowledgement (ACK) paket TCP SYN ACK. Setelah paket TCP SYN ACK diterima dengan baik oleh klien (pengirim), maka klien akan mengirimkan paket ACK sebagai tanda akan dimulainya transaksi

data. Dalam serangan ini, sebuah host akan menerima paket TCP SYN dalam jumlah yang sangat banyak secara terus menerus sehingga menjadi sibuk dan berakibat macetnya sistem. Ilustrasi dapat dilihat pada Gambar 2.17.



Gambar 2.17 Ilustrasi *SYN Attack*

c. *Teardrop Attack*

Serangan yang dilakukan dengan mengeksploitasi proses *disassembly-reassembly* paket data. Dalam jaringan internet sering kali data harus dipotong menjadi paket yang lebih kecil untuk menjamin reliabilitas dan proses multiple akses jaringan. Pada proses pemotongan data paket yang normal, setiap potongan diberi informasi offset data yang berbunyi *"Potongan byte ini merupakan potongan 600 byte dari total 800 byte paket yang dikirimkan"*. Program teardrop memanipulasi potongan data sehingga terjadi *overlapping* antara paket yang diterima *receiver* setelah potongan-potongan paket disusun kembali.

d. *Ping Floods*

Serangan yang dilakukan penyerang yang mencoba untuk melumpuhkan komputer atau sebuah layanan yang terhubung dengan jaringan yang ditargetkan dengan mengirimkan paket yang cacat atau

dengan ukuran besar dan jumlah yang banyak dengan memanfaatkan perintah ping sederhana.

e. Application Level Attack

Serangan dengan jenis ini merupakan serangan yang menargetkan salah satu aplikasi atau layanan yang ada pada sebuah server, seperti aplikasi layanan e-mail, http, ftp, dll.

2. *Port Scanning*

Metode *port scanning* biasanya digunakan oleh penyerang untuk mengetahui port apa saja yang terbuka dalam sebuah sistem jaringan komputer (Nugroho, 2005:11). *Port scanning* dilakukan sebagai tahap awal sebuah serangan. Untuk dapat melakukan penyerangan, seorang penyusup perlu mengetahui aplikasi apa saja yang berjalan dan siap menerima koneksi dari lokasinya berada.

3. *Packet Sniffing*

Packet Sniffing adalah sebuah metode serangan dengan cara mendengarkan seluruh paket yang lewat pada sebuah media komunikasi, baik itu media kabel maupun radio (Nugroho, 2005:14). Setelah paket-paket yang lewat itu didapatkan, paket-paket tersebut kemudian disusun ulang sehingga data yang dikirimkan oleh sebuah pihak dapat dicuri oleh pihak yang tidak berwenang.

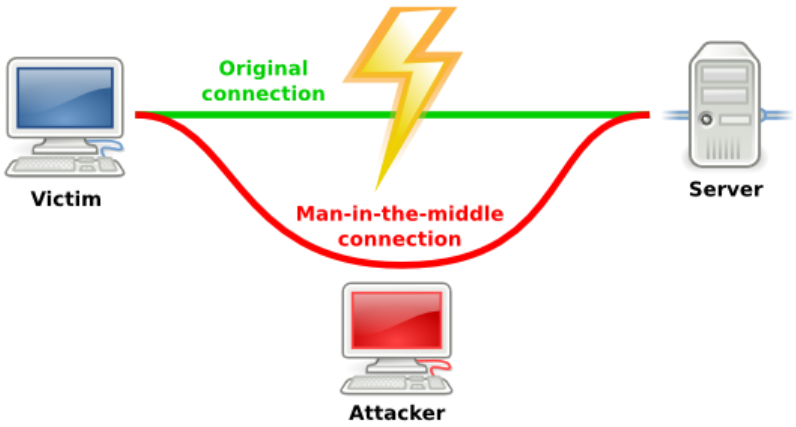
Hal ini dapat dilakukan karena pada dasarnya semua koneksi *ethernet* adalah koneksi yang bersifat *broadcast*, dimana semua *host* dalam sebuah kelompok jaringan akan menerima paket yang dikirimkan oleh sebuah *host*. Pada keadaan normal, hanya *host* yang menjadi tujuan paket yang memproses

paket tersebut sedangkan *host* yang lainnya akan mengacuhkan paket-paket tersebut. Namun pada keadaan tertentu, sebuah *host* bisa merubah konfigurasi sehingga *host* tersebut akan memproses semua paket yang dikirimkan oleh *host* lainnya.

4. *IP Spoofing*

IP Spoofing adalah sebuah model serangan yang bertujuan untuk menipu seseorang. Serangan ini dilakukan dengan cara mengubah alamat asal sebuah paket, sehingga dapat melewati perlindungan *firewall* dan menipu *host* penerima data (Nugroho, 2005:15). Hal ini dapat dilakukan karena pada dasarnya alamat IP asal sebuah paket dituliskan oleh sistem operasi *host* yang mengirimkan paket tersebut.

Salah satu bentuk serangan yang memanfaatkan metode *IP Spoofing* adalah *"man-in-the-middleattack"*. Pada serangan ini, penyerang akan berperan sebagai orang ditengah antara dua pihak yang sedang berkomunikasi. Misalkan ada dua pihak yaitu pihak A dan pihak B lalu ada penyerang yaitu C. Setiap kali A mengirimkan data ke B, data tersebut akan dicegat oleh C, lalu C akan mengirimkan data buatannya sendiri ke B, dengan menyamar sebagi A. Paket balasan dari B ke A juga dicegat oleh C yang kemudian kembali mengirimkan data 'balasan' buatannya sendiri ke A. Dengan cara ini, C akan mendapatkan seluruh data yang dikirimkan antara A dan B, tanpa diketahui oleh A maupun B. Gambar 2.18 mengilustrasikan serangan *man-in-the-middleattack.*



Gambar 2.18 Ilustrasi *Man-in-the-MiddleAttack*

2.1.6 Jenis Serangan Jaringan Komputer di Universitas Negeri Jakarta

Berdasarkan *log* dari *firewall* yang dimiliki UNJ, pada tanggal 28 April 2014 terjadi aktivitas tidak wajar pada jaringan DMZ dimana terdapat *server* yang melakukan *generate traffic* dengan prinsip *DoS* dengan target alamat IP 39.50.197.165 yang dimiliki oleh perusahan telekomunikasi yang berada di Pakistan sebanyak 1769 insiden. Serangan tersebut menggunakan salah satu teknik serangan yang terdapat pada jenis serangan *DoS* yaitu *teardrop attack*, dimana serangan ini mengeksploitasi proses *disassembly-reassembly* paket data yang dikirim yang menyebabkan terjadinya *overlapping* terhadap paket data yang diterima yang dapat menyebabkan komputer penerima mengalami *crash/hang* saat memproses paket tersebut. Tabel 2.2 menunjukkan beberapa baris *log firewall* yang mendeteksi adanya serangan tersebut.

Tabel 2.2 *Log Firewall*

Date	Time	Level	Description
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:60042 to 39.50.197.165:14495 (zone DMZ, int ethernet0/1). Occurred 23 times.
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:6461 to 39.50.197.165:61771 (zone DMZ, int ethernet0/1). Occurred 23 times.

Tabel 2.2 ***Log Firewall***

28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:30482 to 39.50.197.165:26532 (zone DMZ, int ethernet0/1). Occurred 18 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:25360 to 39.50.197.165:29816 (zone DMZ, int ethernet0/1). Occurred 18 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:42412 to 39.50.197.165:50715 (zone DMZ, int ethernet0/1). Occurred 18 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:54630 to 39.50.197.165:59485 (zone DMZ, int ethernet0/1). Occurred 25 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:24646 to 39.50.197.165:23829 (zone DMZ, int ethernet0/1). Occurred 25 times.
28/04 /2014	11:06: 14	emer	Teardrop attack! From 192.168.4.249:63515 to 39.50.197.165:37043 (zone DMZ, int ethernet0/1). Occurred 25 times.
28/04 /2014	11:06: 14	emer	Teardrop attack! From 192.168.4.249:57392 to 39.50.197.165:24485 (zone DMZ, int ethernet0/1). Occurred 18 times.
28/04 /2014	11:06: 14	emer	Teardrop attack! From 192.168.4.249:57522 to 39.50.197.165:41117 (zone DMZ, int ethernet0/1). Occurred 18 times.

2.1.7 Intrusion Detection System

Intrusion didefinisikan sebagai kegiatan yang bersifat *anomaly, incorrect, inappropriate* yang terjadi di jaringan atau di host tersebut (Gondohanindijo, 2012:40). Sedangkan *Intrusion detection* atau deteksi intrusi adalah sebuah proses pemantauan peristiwa yang terjadi dalam sistem komputer atau jaringan dan menganalisanya untuk kemungkinan terjadinya peristiwa yang merupakan pelanggaran atau ancaman terhadap kebijakan keamanan komputer atau jaringan (Scarfone dan Mell, 2007:2-1).

IDS (*Intrusion Detection System*) adalah sebuah aplikasi perangkat lunak atau perangkat keras yang dapat mendeteksi aktivitas yang mencurigakan dalam sebuah sistem atau jaringan (Scarfone dan Mell, 2007:2-1).

Ada beberapa cara di mana sistem IDS dapat dikategorikan. Secara umum IDS dapat dikategorisasikan sebagai berikut (Easttom, 2012:188):

1. *Misuse Detection vs Anomaly Detection*

   Misuse detection atau deteksi penyalahgunaan bekerja dengan cara menganalisis informasi yang dikumpulkan dan membandingkannya dengan sebuah database yang berisi *signature* serangan yang telah didokumentasikan. Sedangkan *anomaly detection* atau deteksi anomali melibatkan perangkat lunak sebenarnya yang bekerja untuk mendeteksi upaya penyusupan dan melaporkannya kepada administrator.

2. *Passive Systems vs Reactive Systems*

   Pada *passive system*, IDS mendeteksi pelanggaran keamanan, melakukan log informasi, dan memberikan sinyal peringatan. Sedangkan pada reactive systems, IDS merespon aktivitas mencurigakan dengan log off pengguna atau memprogram ulang firewall untuk memblokir lalu lintas jaringan dari yang diduga sebagai sumber berbahaya.

3. *Host-Based IDS (HIDS) vs Network-Based IDS (NIDS)*

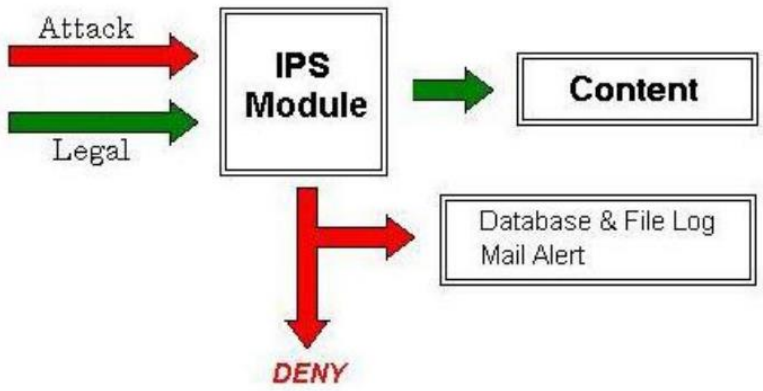
   HIDS bekerja pada host yang akan dilindungi. IDS jenis ini dapat melakukan berbagai macam tugas untuk mendeteksi serangan-serangan yang dilakukan pada host tersebut. Keunggulan HIDS adalah pada tugas-tugas yang berhubungan dengan keamanan file, misalnya ada tidaknya file yang telah

diubah atau usaha untuk mendapatkan akses ke file-file yang sensitif (Arius, 2006:10).

Di lain sisi, NIDS (Network-Based IDS) biasanya berupa suatu mesin yang khusus dipergunakan untuk melakukan monitoring terhadap seluruh segmen dari jaringan. NIDS akan mengumpulkan paket-paket data yang terdapat pada jaringan dan kemudian menganalisisnya serta menentukan apakah paket-paket tersebut berupa suatu paket yang normal atau suatu serangan atau berupa aktivitas yang mencurigakan (Arius, 2006:10).

2.1.8 Intrusion Prevention System

IPS (*Intrusion Prevention System*) merupakan sebuah software yang memiliki semua kemampuan IDS dan juga dapat menghentikan insiden yang terjadi (Scarfone dan Mell, 2007:2-1). Dengan kata lain, IPS merupakan pengembangan dari IDS dengan menambahkan beberapa komponen lainnya seperti *firewall* atau yang lainnya untuk bekerja sama dalam mencegah terjadinya penyusupan. Gambar 2.19 menunjukkan alur kerja IPS.



Gambar 2.19 Intrusion Prevention System

Ada beberapa metode IPS (*Intrusion Prevention System*) dalam melakukan kebijakan apakah paket data yang lewat layak masuk atau keluar dalam jaringan tersebut, antara lain:

1. *Signature-Based Detection*

   Signature adalah pola yang sesuai dengan ancaman yang dikenal. Deteksi berbasis signature adalah proses membandingkan signature terhadap peristiwa yang diamati untuk mengidentifikasi kemungkinan terjadinya insiden. Deteksi berbasis signature sangat efektif dalam mendeteksi ancaman yang dikenal tetapi sebagian besar tidak efektif dalam mendeteksi ancaman yang sebelumnya tidak diketahui.

2. *Anomaly-Based Detection*

   Deteksi berbasis Anomali adalah proses membandingkan definisi tentang apakah aktivitas dianggap normal terhadap peristiwa yang diamati untuk mengidentifikasi penyimpangan yang signifikan. Sebuah IPS menggunakan deteksi berbasis anomali memiliki profil yang mewakili perilaku normal seperti halnya pengguna, host, koneksi jaringan, atau aplikasi. Profil dikembangkan dengan memantau karakteristik aktivitas saat sedang berjalan normal selama periode waktu. Misalnya, profil untuk jaringan menunjukkan bahwa aktivitas Web terdiri dari rata-rata 13% dari batasan bandwidth jaringan Internet selama jam hari kerja biasa. IPS kemudian menggunakan metode statistik untuk membandingkan karakteristik kegiatan saat ini untuk batas yang berkaitan dengan profil, seperti mendeteksi ketika aktivitas Web yang menggunakan lebih banyak bandwidth dari yang biasanya terjadi dalam keadaan normal dan mengambil tindakan sesuai dengan kebijakan yang

diberikan oleh administrator sebelum IPS dijalankan. Profil dapat dikembangkan untuk banyak atribut perilaku, seperti jumlah rata-rata e-mail yang dikirim oleh pengguna, jumlah usaha login yang gagal untuk host, dan tingkat penggunaan prosesor untuk host dalam jangka waktu tertentu. Manfaat utama dari metode pendeteksian berbasis anomali adalah bahwa metode ini dapat menjadi sangat efektif dalam mendeteksi ancaman yang sebelumnya tidak diketahui. Misalnya, bahwa komputer terinfeksi dengan malware tipe baru. Malware bisa mengkonsumsi sumber daya processor pada komputer, mengirim sejumlah besar e-mail, melakukan sejumlah besar koneksi pada jaringan, dan melakukan perilaku lain yang akan secara signifikan berbeda dari profil yang ditetapkan oleh administrator.

2.1.9 Snort

Snort merupakan *NIDS (Network-Based Intrusion Detection System)* dengan sumber terbuka yang mampu melakukan analisis lalu lintas secara *real-time* dan paket *logging* pada jaringan. Hal ini meliputi analisis protokol, *content searching/matching*, dan dapat digunakan untuk mendeteksi berbagai serangan dan penyusupan.

Michael P. Brennan (2002) dalam *paper*nya yang berjudul "*Using Snort For a Distributed Intrusion Detection System*" menjelaskan keuntungan dari penggunaan snort antara lain kemudahannya dalam melakukan konfigurasi dan penggunaan *rules* yang fleksibel dimana apabila terdapat sebuah serangan yang baru dapat dengan mudah menambahkannya dalam *rules database*, serta snort juga memiliki kemampuan dalam menganalisis paket data mentah yang membuatnya menjadi salah satu IDS terbaik. Selain itu, jika dibandingkan dengan

software sejenis seperti *bro*, *snort* memiliki kemampuan pencocokan *rules* dengan serangan yang lebih efisien dibandingkan software *bro* (Moya, 2008:70-71).

2.1.9.1 Komponen Snort

Snort dibagi ke dalam beberapa komponen. Komponen ini bekerjasama untuk mendeteksi serangan yang berbeda dan untuk menghasilkan keluaran pada format yang diinginkan pada sistem deteksi. IDS berbasiskan Snort umumnya terdiri dari komponen:

1. Dekoder Paket

   Dekoder paket mengambil paket dari beberapa jenis jaringan yang berbeda dari antarmuka jaringan dan mempersiapkan paket untuk di proses atau di kirim menuju *detection engine*. Antarmukanya dapat berupa Ethernet, SLIP, PPP dan yang lain.

2. *Preprocessor*

   Preprocessor merupakan komponen atau *plug-ins* yang dapat digunakan pada snort untuk menyusun atau mengubah paket data sebelum detection engine melakukan beberapa operasi untuk mencari tahu jika paket digunakan oleh penyusup. *Preprocessor* pada snort dapat mendekode-kan URL HTTP, men-defragmentasi paket, menggabungkan kembali aliran TCP dan yang lain. Fungsi ini merupakan bagian yang sangat penting pada sistem deteksi intrusi.

3. *Detection Engine*

   Detection engine berfungsi untuk mendeteksi jika terjadi aktifitas penyusup pada paket. Detection engine menggunakan *rules* Snort untuk tujuan ini. *Rules* dibaca ke dalam struktur atau rantai data internal kemudian dicocokkan dengan paket yang ada. Jika paket sesuai dengan rules yang ada,

tindakan akan diambil, jika tidak paket akan diabaikan. Tindakan yang diambil dapat berupa logging paket atau mengaktifkan alert.

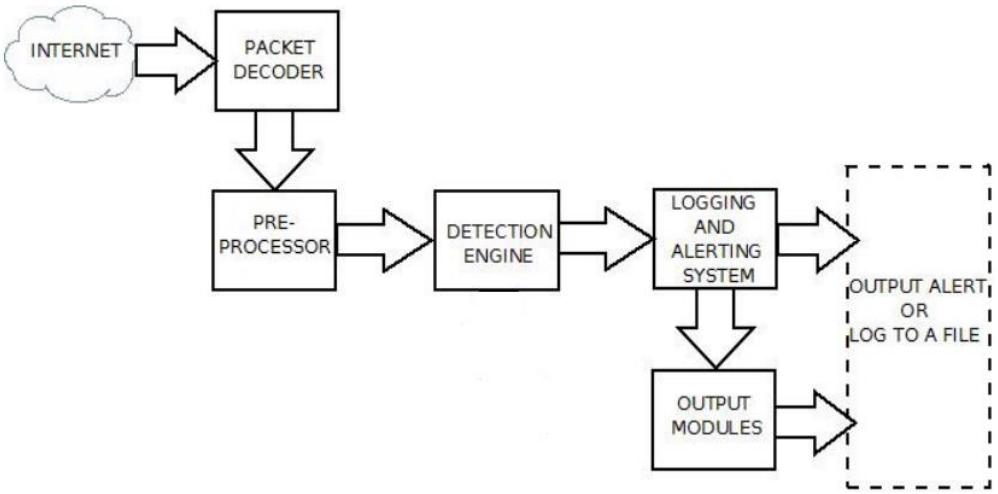
4. Sistem *Log* dan *Alert*

Berdasarkan apa yang ditemukan *detection engine* pada paket, paket dapat digunakan untuk me-*log* kegiatan atau mengaktifkan *alert*, bergantung pada apa yang ditemukan *detection engine* pada paket. *Log* di simpan pada format *file* teks sederhana, file berjenis *tcpdump* atau bentuk yang lain. *File log* disimpan di direktori */var/log/snort* secara *default*. Perintah *snort –l* pada *command line* dapat digunakan untuk memodifikasi lokasi dari *log* dan *alert* yang dihasilkan.

5. Modul Output

Modul Output dapat melakukan beberapa operasi berbeda tergantung bagaimana cara penyimpanan keluaran yang dihasilkan sistem log dan alert dari Snort. Pada dasarnya modul ini mengatur jenis keluaran yang dihasilkan oleh sistem log dan alert. Beberapa keluaran yang dapat dihasilkan tergantung dari konfigurasi, antara lain

a. Logging ke dalam file /var/log/snort/alerts atau file lainnya
b. Mengirimkan pesan ke syslog
c. Logging ke dalam databaseseperti MySQL atau Oracle.
d. Menghasilkan output eXtensible Markup Language(XML)



Gambar 2.20 Komponen Snort

2.1.9.2 Mode Snort

Snort memiliki tiga mode kerja yaitu dapat digunakan sebagai *sniffer mode*, *packet logger*, atau *Intrusion Detection System* (Cisco, 2014).

1. *Sniffer Mode*

   Digunakan hanya untuk melihat paket yang melewati jaringan. Untuk menjalankan snort pada mode ini dapat menggunakan perintah berikut:

   ```
   snort <parameter>
   ```

   Adapun parameter yang dapat digunakan adalah sebagai berikut:

   a. -v , digunakan untuk header TCP/IP paket yang lewat
   b. -d , digunakan untuk melihat isi paket
   c. -e , digunakan untuk melihat header link layer paket seperti ethernet header.

2. *Packet Logger Mode*

   Digunakan untuk mencatat semua paket yang lewat di jaringan untuk di analisa. Untuk menjalankannya dapat menggunakan perintah sebagai berikut:

   ```
   snort -l ./log <parameter>
   ```

Parameter yang dapat ditambahkan adalah sebagai berikut:

a. -h 192.168.1.0/24, paremeter ini digunakan untuk mencatat paket pada host tertentu.

b. -b , digunakan agar file yang di log dalam format binary, bukan ASCII.

Untuk membaca file log dapat, dilakukan dengan menjalankan perintah `snort –dv` dan ditambahkan parameter `–r` nama file log, seperti:

```
snort –dv –r packet.log
snort –dvr packet.log icmp
```

3. *Intrusion Detection System Mode*

Digunakan sebagai *Instrusion Detection System*, untuk mengaktifkan mode ini dengan menambahkan perintah pada *snort* untuk membaca file konfigurasi dengan cara –c nama-file-konfigurasi.conf. Secara default isi file ini sudah diset dalam file *snort.conf* yang termasuk dalam paket *snort*. Berikut ini adalah beberapa perintah yang dapat digunakan untuk mengaktifkan snort sebagai IDS, antara lain

```
./snort -dev -l ./log -h 192.168.1.0/24 -c snort.conf
./snort -d -h 192.168.1.0/24 -l ./log -c snort.conf
```

Untuk melakukan deteksi penyusup snort harus melakukan *logging* paket yang lewat dapat menggunakan perintah –l nama-file-logging, atau membiarkan snort menggunakan default file logging-nya di directory */var/log/snort*. Kemudian menganalisa catatan logging paket yang ada sesuai dengan isi perintah *snort.conf*. Ada beberapa tambahan perintah yang akan membuat proses deteksi menjadi lebih efisien, mekanisme pemberitahuan alert di Linux dapat di atur dengan perintah –A sebagai berikut:

a. -A fast, mode alert yang cepat berisi waktu, berita, IP & port tujuan.

b. -A full, mode alertdengan informasi lengkap.

c. -A unsock, mode alert ke unix socket.

d. -A none, mematikan mode alert.

Untuk mengirimkan *alert* ke *syslog* UNIX kita bisa menambahkan switch –*s*, seperti tampak pada beberapa contoh di bawah ini:

```
snort –c snort.conf –l ./log –s –h 192.168.0.0/24
snort –c snort.conf –s –h 192.168.0.0/24
```

Untuk mengirimkan alert biner ke workstation Windows, dapat digunakan perintah di bawah ini :

```
snort –c snort.conf –b –M WORKSTATIONS
```

Agar snort beroperasi secara langsung setiap kali workstationatau server diboot, kita dapat menambahkan ke file */etc/rc.d/rc.local* perintah di bawah ini:

```
/usr/local/bin/snort -d -h 192.168.0.0/24 -c
/root/snort/snort.conf –A full –s –D
```

Atau

```
/usr/local/bin/snort –d –c /root/snort/snort.conf –A full
–s –D
```

dimana –D adalah parameter yang mengatur agar snort bekerja sebagai daemon.

2.1 Kerangka Berpikir

Untuk mengimplementasikan sebuah sistem keamanan jaringan dengan *Intrusion Prevention System (IPS)* menggunakan snort, harus berdasarkan prosedur yang diawali dengan melakukan identifikasi masalah yang telah dijelaskan pada Bab I yang bertujuan untuk mengangkat masalah yang akan di teliti.

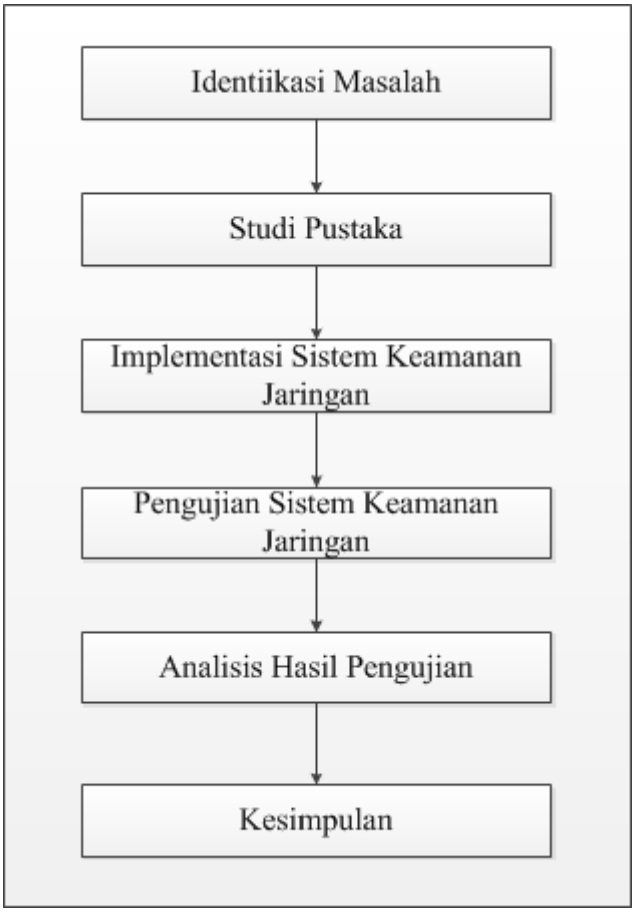
Langkah selanjutnya adalah melakukan studi pustaka yang bertujuan untuk memperkuat materi pembahasan sekaligus menjadi dasar untuk menggunakan teori-teori tertentu dalam menyelesaikan masalah.

Langkah selanjutnya adalah implementasi sistem keamanan jaringan, yang meliputi pengumpulan data dan analisis terhadap kondisi jaringan yang terdapat di PUSTIKOM UNJ. Dilanjutkan dengan melakukan instalasi sistem operasi dan perangkat lunak snort serta mengkonfigurasinya pada sebuah PC yang akan dijadikan sensor yang akan memantau lalu lintas jaringan.

Setelah implementasi dilakukan, perlu dilakukan pengujian terhadap sistem keamanan jaringan yang telah dibuat. Tujuannya adalah apakah sistem keamanan jaringan tersebut mampu menangani setiap serangan yang dilakukan. Jika hasil pengujian belum dapat menunjukan bahwa sistem keamanan jaringan tersebut mampu menangani setiap serangan yang dilakukan, maka perlu dilakukan pengecekan dan perbaikan terhadap langkah-langkah pada tahap implementasi.

Jika data dari hasil pengujian sistem keamanan jaringan tersebut sudah sesuai dengan yang diharapkan. Langkah selanjutnya adalah melakukan analisis terhadap hasil pengujian, apakah sistem keamanan ini mampu menghalau setiap serangan yang datang, kemudian juga perlu dianalisi apakah sistem keamanan ini menggangu kinerja jaringan atau tidak.

Yang terakhir adalah penarikan kesimpulan berdasarkan analisis yang telah dilakukan pada tahap sebelumnya. Kerangka berpikir pada penelitian ini diilustrasikan pada Gambar 2.21



Gambar 2.21 Bagan Kerangka Berpikir

BAB III
METODOLOGI PENELITIAN

3.1 Tempat dan Waktu Penelitian

Penelitian dilakukan di Pusat Teknologi Informasi dan Komunikasi (PUSTIKOM) Universitas Negeri Jakarta. Waktu penelitian dilakukan sejak bulan April 2014 hingga Desember 2014.

3.2 Metode Penelitian

Metode penelitian yang digunakan pada penelitian ini adalah metode *Research and Development (R&D)*.Metode *Research and Development* ini merupakan metode penelitian yang digunakan untuk menghasilkan produk tertentu dan menguji keefektifan produk tersebut.

Diadaptasi dari Borg dan Gall, Sugiyono merumuskan metode *research and development* ke dalam 10 langkah (Sugiyono, 2011:298), yaitu:

1. Mengidentifikasi masalah
2. Pengumpulan data
3. Mengembangkan rancangan sistem
4. Validasi rancangan sistem
5. Merevisi rancangan sistem
6. Menguji rancangan sistem pada skala kecil
7. Merevisi rancangan sistem
8. Menguji rancangan sistem pada skala besar
9. Merevisi rancangan sistem
10. Menerapkan rancangan sistem

3.3 Instrumen Penelitian

Penelitian ini menggunakan instrument yang berupa perintah uji koneksi dan program aplikasi, antara lain:

1. Perintah *ping* dengan mengirimkan paket berukuran besar yang dapat dieksekusi dari *terminal* atau *command prompt*
2. Program aplikasi LOIC

3.4 Perangkat Penelitian

Dalam penelitian ini perangkat yang digunakan terdiri dari:

1. Perangkat Keras
    a. Sebuah komputer yang bertindak sebagai sistem keamanan dengan spesifikasi sebagai berikut:
        - Processor Intel® Pentium™ D CPU @ 2.80GHz
        - Memori 2GB DDR2
        - HardDisk 40GB
        - 2 buah Gigabit Ethernet Card

        Adapun sistem operasi yang digunakan pada sistem keamanan ini adalah linux dengan distro CentOS 6.5 x86_64
    b. Switch yang digunakan untuk menghubungkan beberapa perangkat komputer ke dalam jaringan
    c. Kabel UTP dan konektor RJ-45 sebagai media penghubung antar perangkat dalam jaringan
2. Perangkat Lunak
    a. Paket snort-2.9.6.2.tar.gz
    b. Paket daq-2.0.2-1.centos6.x86_64.rpm

c. Paket barnyard2-2-1.13.tar.gz

d. Paket base-1.4.5.tar.gz

e. Paket pulledpork-0.7.0.tar.gz

f. Dan beberapa paket library pendukung lainnya.

3.5 Prosedur Penelitian dan Pengembangan

Prosedur penelitian dan pengembangan sistem keamanan dengan IPS tersusun sebagai berikut:

1. Melakukan pengumpulan data

   Pada tahap pertama ini, peneliti melakukan wawancara kepada *administrator* jaringan dan observasi kondisi jaringan yang ada di PUSTIKOM Universitas Negeri Jakarta, serta mengumpulkan data yang terkait dengan *log* atau data serangan yang pernah terjadi.

2. Menganalisis sistem

   Kemudian melakukan analisis kebutuhan, berupa penggunaan *software* apa saja yang dibutuhkan oleh sistem agar dapat melakukan pemblokiran dan dapat menampilkan log dari serangan yang terjadi.

3. Merancang desain sistem yang akan dikembangkan

   Setelah mendapatkan data yang diperlukan dan dari hasil menganalisa kebutuhan apa saja pada sistem, peneliti mendesain topologi yang akan digunakan agar sistem yang dibuat dapat secara signifikan melakukan pemblokiran terhadap serangan yang terjadi.

4. Merancang sistem

   Setelah itu merancang sistem dengan melakukan konfigurasi yang diperlukan agar sistem dapat bekerja sesuai dengan yang diharapkan.

5. Melakukan uji fungsionalitas sistem

   Pada tahap ini, sistem yang telah dikonfigurasi diuji coba fungsionalitasnya apakah sistem bekerja dengan semestinya.

6. Melakukan uji validitas sistem

   Jika sistem sudah bekerja sesuai dengan yang diharapkan, kemudian dilakukan uji validitas terhadap parameter yang digunakan untuk melakukan pemblokiran terhadap serangan yang terjadi.

7. Menganalisis data hasil uji validitas

   Dari hasil uji validitas yang telah dilakukan, data dianalisis apakah parameter yang digunakan untuk melakukan pemblokiran telah berhasil atau belum dalam menghalau serangan.

8. Merevisi rancangan sistem

   Apabila parameter yang digunakan belum dapat melakukan pemblokiran terhadap serangan yang dilakukan, maka perlu adanya perbaikan pada parameter yang digunakan hingga didapatkan hasil yang sesuai dengan harapan.

9. Melakukan uji pada jaringan skala kecil

   Setelah melakukan perbaikan terhadap parameter yang digunakan hingga didapatkan hasil yang diharapkan, kemudian sistem diuji coba pada sebuah jaringan dengan skala kecil.

10. Menganalisis data hasil uji pada jaringan skala kecil

    Dari data hasil uji yang telah dilakukan, data tersebut dianalisis apakah pada saat diuji coba pada jaringan dengan skala yang kecil masih terdapat serangan yang lolos atau tidak.

11. Merevisi rancangan sistem

    Jika saat uji coba pada jaringan skala kecil masih terdapat serangan yang lolos dari sistem keamanan yang dirancang, maka akan dilakukan perbaikan kembali hingga mampu melakukan pemblokiran terhadap serangan yang dilakukan pada jaringan.

12. Melakukan uji pada jaringan skala besar

    Setelah dilakukan perbaikan dan dianggap layak untuk diuji coba pada sistem dengan skala yang lebih besar, maka sistem keamanan yang dirancang ditempatkan diantara jaringan DMZ UNJ untuk dilakukan uji coba pada jaringan tersebut.

13. Menganalisis data hasil uji pada jaringan skala besar

    Dari data hasil uji yang telah didapatkan pada saat uji coba sistem di jaringan dengan skala yang lebih besar yaitu pada jaringan DMZ UNJ, data tersebut kemudian dianalisis apakah setiap serangan yang dilakukan berhasil diblokir/dihalau oleh sistem yang dirancang.

14. Merevisi rancangan sistem

    Jika pada saat uji coba pada jaringan dengan skala besar masih ada beberapa serangan yang belum dapat diblokir, maka perlu dilakukan perbaikan kembali. Setelah dilakukan beberapa kali perbaikan, hingga sistem keamanan yang dirancang sudah memenuhi kriteria yang diinginkan, kemudian dilakukan penyempurnaan sistem.

15. Mengimplementasikan sistem pada jaringan DMZ UNJ

    Setelah keseluruhan proses revisi sistem telah selesai dilakukan, maka sistem siap untuk diimplementasikan pada jaringan DMZ UNJ.

3.6 Hasil Pengujian Sistem Keamanan IPS

Kriteria hasil pengujian terhadap sistem keamanan IPS yang didapatkan dari sensor didasarkan pada:

1. Hasil dari sensor IPS dapat menampilkan informasi mengenai aktivitas serangan yang terjadi, meliputi alamat IP sumber, alamat IP tujuan, jenis serangan dan protokol yang digunakan.
2. Sistem keamanan IPS dapat melakukan pemblokiran terhadap serangan yang dijalankan menggunakan instrument penelitian yang sudah didefinisikan sebelumnya.

3.7 Pengujian Sistem Keamanan IPS

Pengujian sistem keamanan jaringan menggunakan IPS dilakukan dari jaringan DMZ UNJ. Adapun spesifikasi komputer yang digunakan untuk menyerang meliputi processor Intel i5-2450M 2.5GHz, ram 8GB dengan kapasitas harddisk 500GB yang terhubung dengan switch yang berada di jaringan DMZ UNJ.

Pengujian dilakukan menggunakan perangkat lunak yang sudah didefinisikan pada bagian instrumen penelitian yaitu penggunaan perintah ping melalui *Command Prompt* yang digunakan untuk melakukan serangan *Ping Flood* dan LOIC untuk melakukan serangan *DoS* dengan target *service* HTTP.

3.7.1 Prosedur Pengujian dengan Perintah PING

Untuk mengetahui apakah sistem keamanan IPS dapat melakukan *blocking* serangan *Ping Flood*, dapat menggunakan utilitas ping yang terdapat pada sistem operasi dengan cara sebagai berikut:

1. Buka *Command Prompt*, kemudian ketik perintah baris berikut

ping <target> -l <size> -n <count> / -t

2. Setelah itu tekan Enter untuk memulai.

Dari perintah baris di atas untuk <*target*> masukan alamat IP atau *domain* dari suatu *host/server*, kemudian <*size*> isikan besaran paket yang akan dikirim dengan jumlah maksimal 65500 bytes dan <*count*> masukan jumlah *request* yang diinginkan atau menggunakan *parameter* –t untuk melakukan *request* hingga *host* yang menjadi *target* berhenti membalas. Misal: ping google.com –l 35 –n 1000 atau ping google.com –l 35 -t

3.7.2 Prosedur Pengujian dengan LOIC

Pengujian selanjutnya menggunakan LOIC yang merupakan perangkat lunak yang dapat melakukan *generate traffic* dengan jumlah yang banyak. Adapun langkah-langkahnya sebagai berikut:

1. Buka *program LOIC* yang telah disiapkan
2. Pada jendela yang muncul, masukan *URL* atau alamat IP *target* yang akan diserang, kemudian klik *Lock on*.
3. Pada opsi method pilih HTTP untuk mengincar *service* HTTP pada *host target*.
4. Setalah itu klik tombol *IMMA CHARGIN MAH LAZER* untuk memulai

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

4.1 Hasil Penelitian

4.1.1 Observasi dan Analisis Terhadap Kondisi Jaringan Komputer UNJ

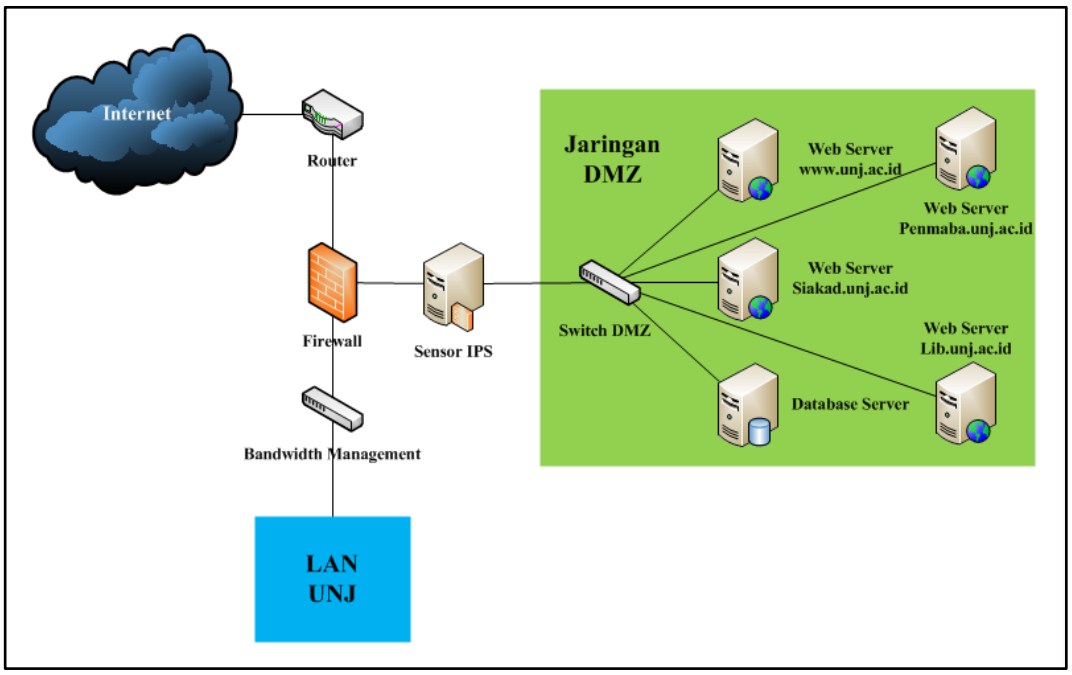
Sebagai langkah awal, peneliti melakukan observasi terhadap kondisi jaringan yang terdapat di PUSTIKOM UNJ. Observasi ini meliputi pengumpulan data terhadap serangan yang pernah terjadi dan pengamatan langsung ke ruang server PUSTIKOM UNJ, tempat dimana sensor IPS akan diletakan. Dalam pengumpulan data berkaitan dengan serangan yang pernah terjadi peneliti melakukan wawancara dengan *administrator* jaringan UNJ. Dari hasil wawancara peneliti memperoleh informasi yang lebih detail, yang meliputi jenis dan sumber serangan yang pernah terjadi. Dari informasi tersebut kebanyakan serangan yang pernah terjadi bersumber dari sebuah server yang berada pada jaringan DMZ UNJ dengan menggunakan teknik DoS.

Selain itu peneliti juga mendiskusikan hasil pengamatan langsung kepada administrator mengenai penempatan sensor IPS pada jaringan UNJ. Dari hasil pengamatan langsung, peneliti mengetahui bahwa jaringan komputer di UNJ berpusat pada sebuah firewall yang membagi 3 jaringan utama, yaitu jaringan internet (WAN), jaringan DMZ dan jaringan lokal (LAN).

Setelah melakukan observasi kemudian peneliti melakukan analisis berdasarkan data yang telah diperoleh dari hasil wawancara, diskusi dan pengamatan langsung, yang akhirnya memutuskan untuk meletakan sensor IPS diantara *firewall* dan *switch* yang terhubung pada jaringan DMZ.

4.1.2 Penempatan Sensor IPS Pada Jaringan DMZ UNJ

Sensor IPS diletakkan diantara switch yang berada di jaringan DMZ dengan *firewall*. Sensor IPS dikonfigurasi menjadi sebuah *bridge* yang dihubungkan menggunakan kabel UTP CAT 6 dengan susunan straight dimana nantinya setiap paket data yang lewat akan dilakukan pemeriksaan oleh sensor tersebut. Untuk lebih jelasnya dapat dilihat pada Gambar 4.1.



Gambar 4.1 Penempatan Sensor IPS pada Jaringan DMZ Universitas Negeri Jakarta

4.1.3 Instalasi Linux CentOS 6.5 x86_64

1. Masukkan DVD *bootable* Linux CentOS 6.5 x86_64
2. Masuk ke BIOS (*Basic Input Output System*), kemudian ubah *1st boot device* pada komputer menjadi *CD/DVD*. Berikutnya simpan konfigurasi lalu keluar dari BIOS.

3. Komputer akan *restart* dan mulai *booting* dari DVD. Pada saat muncul tulisan *Press any key to boot from CD or DVD*, tekan sembarang tombol pada keyboard untuk memuali proses *booting* melalui *CD/DVD*.
4. Pada jendela instalasi yang muncul pertama kali, pilih *Install or upgrade an existing system*, lalu tekan Enter.
5. Pada pilihan pemeriksaan media, pilih *Skip* untuk melanjutkan ke tahap berikutnya. Lalu klik Next.
6. Pada pilihan bahasa, pilih *English (English).* Lalu klik *Next*.
7. Pada pilihan standar keyboard yang digunakan sistem, pilih *U.S. English.* Lalu klik *Next*.
8. Pada pilihan *Device Type* yang akan di instal, pilih *Basic Storage Devices.* Lalu klik *Next.*
9. Pada jendela *pop-up Storage Device Warning* yang muncul, centang *Apply my choice to all devices with undetected partitions or filesystems*, setelah itu klik *Yes, discard any data*.
10. Pada jendela nama hostname, ubah menjadi IPSSVR01. Lalu klik Next.
11. Pada pilihan City & Timezone, pilih Asia/Jakarta. Klik Next.
12. Kemudian masukkan Root Password dan konfirmasi ulang password yang dimasukkan, lalu klik Next.
13. Pada pilihan format harddisk, pilih *Use All Space*, klik *Next*. Kemudian pada *pop-up* yang muncul klik *Write changes to disk.*
14. Kemudian pada pilihan instalasi *software* CentOS, pilih Desktop lalu klik Next.
15. Tunggu hingga proses instalasi selesai hingga komputer melakukan restart.

16. Pada tampilan awal yang menyatakan telah berhasil menginstalasi CentOS, klik *Forward* untuk melanjutkan.
17. Pada halaman persetujuan licensi CentOS, pilih *Yes, I agree to the License Agreement* lalu klik *Forward*.
18. Pada jendela buat *user*, masukkan *username, full name, password* pada kotak *input* yang tersedia, lalu klik *Forward*.
19. Pada jendela pengaturan tanggal dan waktu, atur sesuai dengan kondisi yang ada, kemudian klik Forward.
20. Pada pilihan *Kdump (Kernel Dump)*, centang pilihan *Enable kdump?* , lalu klik *Finish*.
21. Kemudian pada *pop-up* yang muncul klik *Yes*. Lalu sistem akan melakukan reboot dan tunggu hingga muncul tampilan login.

4.1.4 Instalasi dan Konfigurasi Snort

4.1.4.1 Instalasi Library Pendukung Snort

Sebelum melakukan instalasi snort, ada beberapa *library* yang harus di instal terlebih dahulu yang dibutuhkan oleh snort, yaitu *libnet, libdnet-devel, libpcap, libpcap-devel, gcc, make, flex, bison, pcre, pcre-devel, zlib, zlib-devel dan DAQ* yang dapat diinstal menggunakan perintah *yum* dari terminal:

```
[root@IPSSVR01 / ] # yum -y install libnet libdnet-devel
libpcap libpcap-devel gcc make flex bison pcre pcre-devel
zlib zlib-devel
```

Untuk paket *DAQ* instalasi dilakukan secara manual melalui *file source DAQ* yang sebelumnya disimpan di direktori /usr/local/src

```
[root@IPSSVR01 src] # tar xzvf daq-2.0.1.tar.gz
[root@IPSSVR01 src] # cd daq-2.0.1
[root@IPSSVR01 daq-2.0.1] # ./configure
[root@IPSSVR01 daq-2.0.1] # make
[root@IPSSVR01 daq-2.0.1] # make install
```

4.1.4.2 Instalasi Snort

Untuk instalasi snort dapat dilakukan dari *terminal* menggunakan *file source* snort yang telah disimpan di direktori /usr/local/src menggunakan perintah berikut dari terminal:

```
[root@IPSSVR01 src] # tar xzvf snort-2.9.6.2.tar.gz
[root@IPSSVR01 src] # cd snort-2.9.6.2
[root@IPSSVR01 snort-2.9.6.2] # ./configure --enable-ipv6 --
enable-gre --enable-mpls --enable-decoder-preprocessor-rules
--enable-ppm --enable-perfprofiling --enable-zlib --enable-
active-response   --enable-normalizer   --enable-reload   --
enable-react --enable-flexresp3
[root@IPSSVR01 snort-2.9.6.2] # make
[root@IPSSVR01 snort-2.9.6.2] # make install
```

4.1.4.3 Konfigurasi Snort IPS

1. Pengaturan *Bridge* menggunakan perintah berikut dari *terminal:*

   ```
   [root@IPSSVR01 src] # brctl addbr br0
   [root@IPSSVR01 src] # brctl addif br0 eth1
   [root@IPSSVR01 src] # brctl addif br0 eth2
   ```

   Hidupkan *interface bridge* yang telah dibuat, dengan perintah:

   ```
   [root@IPSSVR01 src] # ifconfig br0 up
   ```

2. Pengaturan *DAQ* pada *file snort.conf* yang berada di direktori /etc/snort untuk membukanya dapat menggunakan perintah:

   ```
   [root@IPSSVR01 src] # nano /etc/snort/snort.conf
   ```

   Kemudian cari dan ubah menjadi baris berikut dan pastikan *uncomment* baris tersebut.

   ```
   config daq: afpacket
   config daq_mode: inline
   config policy_mode: inline
   ```

   Untuk lebih lengkapnya, dapat dilihat pada Lampiran 2.

3. Pengaturan *Snort Startup Scripts*

   Buka dan modifikasi file /etc/rc.local dengan menambahkan baris berikut:

   ```
   BRIDGE= /usr/ sbin/brctl
   ```

```
IFCONFIG= /sbin/ifconfig
$BRIDGE addbr br0
$BRIDGE addif br0 eth1
$BRIDGE addif br0 eth2
$IFCONFIG eth1 0.0.0.0 up –arp
$IFCONFIG eth2 0.0.0.0 up –arp
$IFCONFIG br0 0.0.0.0 up –arp
sysctl net.ipv4.ip_forward=0
```

Setelah itu, buka dan modifikasi file /etc/sysconfig/snort menjadi baris berikut:

```
QUEUE=1
INTERFACE=eth1:eth2
```

Kemudian buka dan modifikasi file /etc/init.d/snortd, di bawah baris # *Source the local configuration file* tambahkan baris berikut

```
IFCONFIG=/sbin/ifconfig
```

Lalu cari baris yang berisi kurang lebih seperti di bawah ini

```
# Now to the real heart of the matter
```

Dan tambahkan baris perintah berikut di atasnya

```
if [ "$QUEUE"X = "1X" ]; then
      QUEUE="-Q"
else
      QUEUE=""
fi
```

Di bawah baris yang diidentifikasi di atas, cari *statement case*. Pada baris setelah kata *start)* tambahkan baris berikut

```
$IFCONFIG br0 down
```

Lalu cari 3 *statement* yang meminta *daemon* untuk menjalankan proses snort. Modifikasi masing-masing dengan menambahkan *string* $QUEUE setelah /usr/sbin/snort. Kurang lebih seperti ini

```
Daemon /usr/sbin/snort $QUEUE $ALERTMODE $BINARY_LOG
$NO_PACKET_LOG $DUMP_APP -D $PRINT_INTERFACE -i $i -u
$USER -g $GROUP $CONF -l $LOGDIR/$i $PASS_FIRST
$BPFFILE $BPF
```

Setelah itu, cari stop) pada *statement* case. Di bawah perintah *echo* ke 2 tambahkan perintah `$IFCONFTG br0 up –arp`

Untuk lebih lengkapnya, dapat dilihat pada Lampiran 3.

4.1.5 Instalasi dan Konfigurasi Barnyard2

Barnyard2 merupakan perangkat lunak yang berfungsi untuk mengelola *unified output* yang dibuat oleh snort dan mengubahnya ke bentuk *output* yang diinginkan seperti ASCII, PCAP, atau Database (Sourcefire:51).

1. Lakukan instalasi dari *terminal* menggunakan file source barnyard2 yang telah disimpan di direktori /usr/local/src

```
[root@IPSSVR01 src] # tar xzvf barnyard2-2-1.13.tar.gz
[root@IPSSVR01 src] # cd barnyard2-2-1.13
[root@IPSSVR01 barnyard2-2-1.13] # ./autogen.sh
[root@IPSSVR01 barnyard2-2-1.13] # ./configure --with-
mysql
[root@IPSSVR01 barnyard2-2-1.13] # make
[root@IPSSVR01 barnyard2-2-1.13] # make install
```

2. Setelah itu copy file config barnyard2.conf ke direktori /etc/snort

```
[root@IPSSVR01 barnyard2-2-1.13] # cp /etc/barnyard2
/etc/snort
```

3. Kemudian buat direktori untuk log barnyard2

```
[root@IPSSVR01 barnyard2-2-1.13] # mkdir
/var/log/barnyard2
```

4. Lalu buka file config barnyard2.conf, cari dan modifikasi menjadi baris berikut:

```
config reference_file: /etc/snort/reference.config
config                              classification_file:
/etc/snort/classification.config
config gen_file:  /etc/snort/gen-msg.map
config sid_file: /etc/snort/sid-msg.map
input unified2
config hostname: IPSSVR01
config interface: eth1:eth2
config waldo_file: /var/log/snort/eth1/barnyard2.waldo
config archivedir: /var/log/snort/eth1/archive
config process_new_records_only
```

```
output database: log, mysql, user=snort
password=snortpassword dbname=snort host=localhost
```

5. Kemudian ubah log snort menjadi unified2 pada file snort.conf menjadi:

```
output unified2: filename merged.log, limit 128
```

6. Selanjutnya buat snort *database* dan *user account* MySQL:

```
[root@IPSSVR01] # mysql
mysql> set password for
root@localhost=password("password");
mysql> create database snort;
mysql> grant create, insert, select, delete, update on
snort.* to snort@localhost;
mysql> set password for
snort@localhost=password("password");
mysql> exit;
```

7. Kemudian ikuti perintah berikut untuk membuat *database schemas* untuk snort:

```
[root@IPSSVR01] # cd /usr/local/src/barnyard2-2-
1.13/schemas
[root@IPSSVR01] # mysql –p < create_mysql snort
```

Mysql akan meminta password, masukkan password yang telah dibuat untuk user root.

Konfigurasi lengkapnya dapat dilihat pada Lampiran 4.

4.1.6 Instalasi dan Konfigurasi BASE

BASE *(Basic Analysis and Security Engine)* adalah alat pelaporan dan analisis snort berbasis web yang digunakan untuk menampilkan *log* milik snort agar mudah di analisis dibandingkan membacanya langsung dalam bentuk *file log* (Sourcefire:65). BASE mengumpulkan data dari database yang dibuat oleh barnyard2 dan menampilkannya dalam bentuk web.

Untuk menginstalasinya dapat menggunakan *file source* base yang telah disimpan di direktori /usr/local/src

1. Buka terminal dan masukkan perintah berikut untuk melakukan instalasi paket tambahan yang dibutuhkan BASE:

```
[root@IPSSVR01 src] # yum install -y mysql-server
mysql-devel php-mysql php-pear php-gd httpd
[root@IPSSVR01 src] # pear channel-update pear.php.net
[root@IPSSVR01 src] # pear install Numbers_Roman-1.0.2
[root@IPSSVR01 src] # pear install Image_Canvas-0.3.5
[root@IPSSVR01 src] # pear install Image_Graph-0.8.0
```

2. Lalu *unpack* paket adodb php dan copy ke direktori /var/www/ dari *terminal* menggunakan perintah:

```
[root@IPSSVR01 src] # tar xzvf adodb519.tar.gz
[root@IPSSVR01 src] # cp adodb5 /var/www/
```

3. Kemudian instalasi BASE dari terminal menggunakan perintah:

```
[root@IPSSVR01 src] # tar xzvf base-1.4.5.tar.gz
[root@IPSSVR01 src] # cd base-1.4.5.tar.gz
[root@IPSSVR01 src] # mkdir /var/www/base
[root@IPSSVR01 src] # cp -r base-1.4.5/ /var/www/base
[root@IPSSVR01 src] # cd /var/www/base/
[root@IPSSVR01 src] # cp base_conf.php.dist
base_conf.php
```

4. Setelah itu buka dan modifikasi file *php.ini* untuk mengubah *error reporting level,* menjadi baris berikut:

```
error_reporting = E_ALL & ~E_NOTICE
;error_reporting = E_ALL
```

5. *Restart httpd process* untuk melihat perubahan yang telah dilakukan pada file konfigurasi PHP menggunakan perintah:

```
[root@IPSSVR01 src] # service httpd restart
```

6. Buka BASE melalui *web browser* untuk melakukan konfigurasi dengan mengakses alamat berikut:

```
http://localhost/base
```

7. Pada tampilan awal BASE klik *Continue* untuk melanjutkan

8. Kemudian pada tampilan step 1, pilih bahasa *English* dan arahkan adodb pada direktori /var/www/adodb5 lalu klik *Continue*
9. Lalu pada tampilan step 2, masukkan *database name, database host, database username, dan database password* yang telah dibuat sebelumnya, lalu klik *Continue*
10. Setelah itu pada tampilan step 3, masukkan *admin username, password*, dan full name pengelola BASE, klik *Continue*
11. Pada tampilan step 4, ikuti petunjuk yang menganjurkan membuat *database* yang digunakan oleh BASE dengan mengklik tombol *Create BASE AG* dan klik *Continue*
12. Terakhir, login menggunakan username dan password yang tadi dimasukkan pada step ke 3.

4.1.7 Instalasi dan Konfigurasi Pulledpork

Pulledpork merupakan paket tambahan yang digunakan untuk melakukan *update rules* snort terhadap jenis serangan terbaru (Sourcefire:221). Untuk dapat melakukan *update* dari pulledpork, sebelumnya harus menjadi *member* dan men-*generate code* dari halaman *member* di situs snort.org

Untuk menginstalasinya dapat menggunakan *file source* pulledpork yang telah disimpan di direktori /usr/local/src

1. Buka terminal dan masukkan perintah baris berikut:

```
[root@IPSSVR01 src] # tar xzvf pulledpork-0.7.0.tar.gz
[root@IPSSVR01 src] # cd pulledpork-0.7.0.tar.gz
[root@IPSSVR01 pulledpork-0.7.0] # cp pulledpork.pl
/usr/local/bin
[root@IPSSVR01 pulledpork-0.7.0] # chmod 755
/usr/local/bin/pulledpork.pl
[root@IPSSVR01 pulledpork-0.7.0] # mkdir
/etc/pulledpork
```

```
[root@IPSSVR01 pulledpork-0.7.0] # cp etc/*.conf
/etc/pulledpork
```

2. Setelah itu buka file konfigurasi pulledpork.conf yang ada di direktori /etc/pulledpork. Kemudian cari baris berikut:

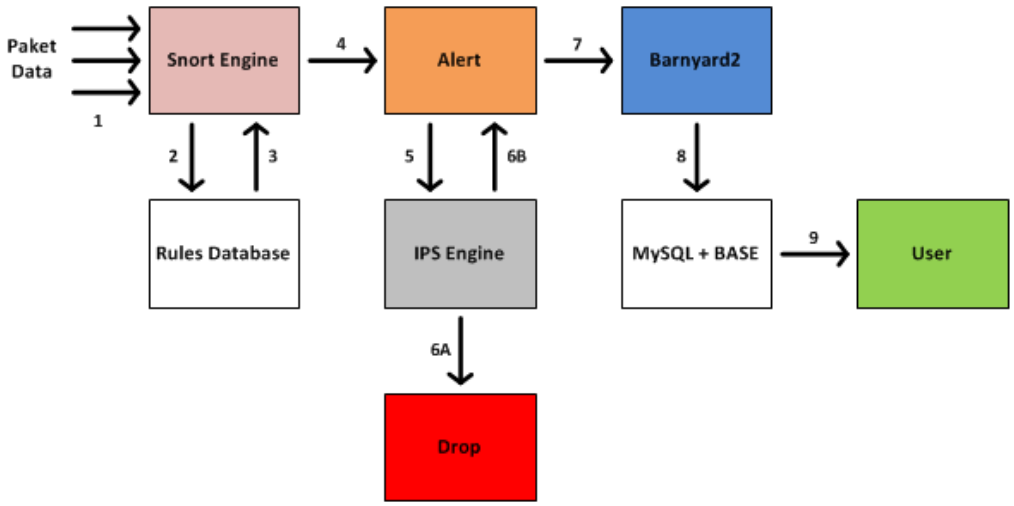
```
rule_url=https://www.snort.org/reg-rules/|snortrules-
snapshot.tar.gz|<oinkcode>
rule_url=https://s3.amazonaws.com/snort-
org/www/rules/community/|community-
rules.tar.gz|Community
rule_url=https://www.snort.org/reg-
rules/|opensource.gz|<oinkcode>
```

3. Pada baris yang terdapat kata <*oinkcode*>, ganti dengan kode yang didapat dari situs snort.org

Untuk konfigurasi secara lengkap dapat dilihat pada Lampiran 5.

4.1.8 Cara Kerja

Snort yang merupakan IDS berbasis jaringan bekerja menggunakan *rule/signature based*, dengan cara menganalisis dan membandingkannya dengan jenis serangan yang sudah diketahui dan terdapat pada *rules* milik snort. Snort menggunakan *tcpdump* untuk mengumpulkan dan menganalisa paket data terhadap jenis serangan yang sudah terdefinisi. Ketika snort dikonfigurasi menjadi sebuah IPS, snort dapat memiliki kemampuan tambahan yaitu melakukan *drop* terhadapa paket yang telah dianalisa dan dibandingkan dengan rules yang dimilikinya. Cara kerja snort sebagai IPS dapat dilihat pada Gambar 4.2.

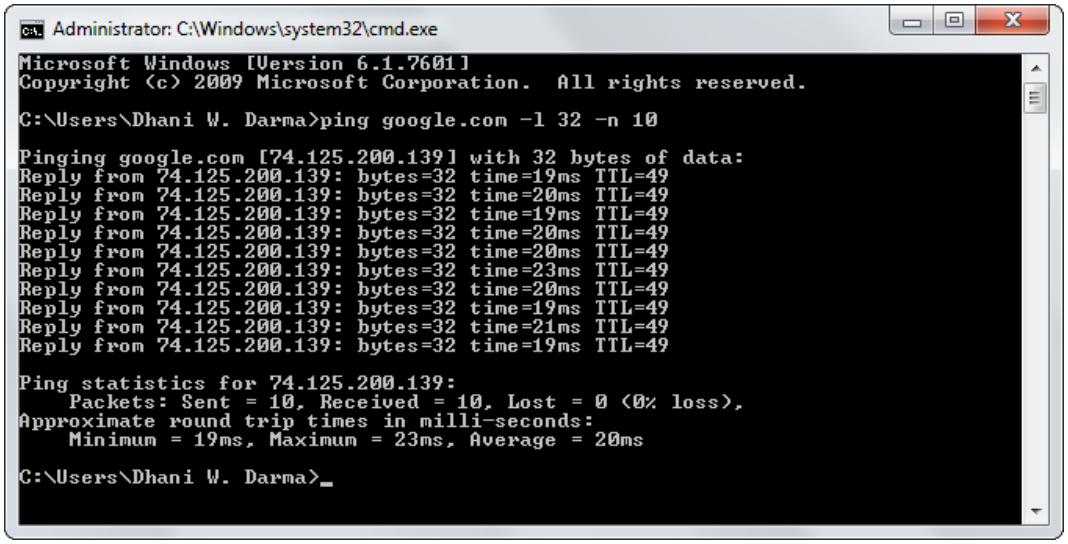


Gambar 4.2 Cara Kerja Sistem Keamanan IPS

Ketika terdapat sebuah paket data yang merupakan serangan, *snort engine* akan menganalisa dan membandingkannya dengan *rules* yang telah dimilikinya kemudian mengirimkan peringatan ke *alert log*, kemudian *IPS engine* membaca *alert* tersebut dan apabila *alert* tersebut memiliki parameter *drop* sesuai dengan yang dikirim oleh *snort engine* maka *IPS engine* akan melakukan *drop* terhadap paket tersebut. Selanjutnya Barnyard2 mengkonversi *alert* yang dibuat oleh *snort engine* ke dalam bentuk *database* yang selanjutnya digunakan oleh BASE untuk menampilkannya dalam bentuk web.

4.1.9 Hasil Pengujian dengan Perintah Ping

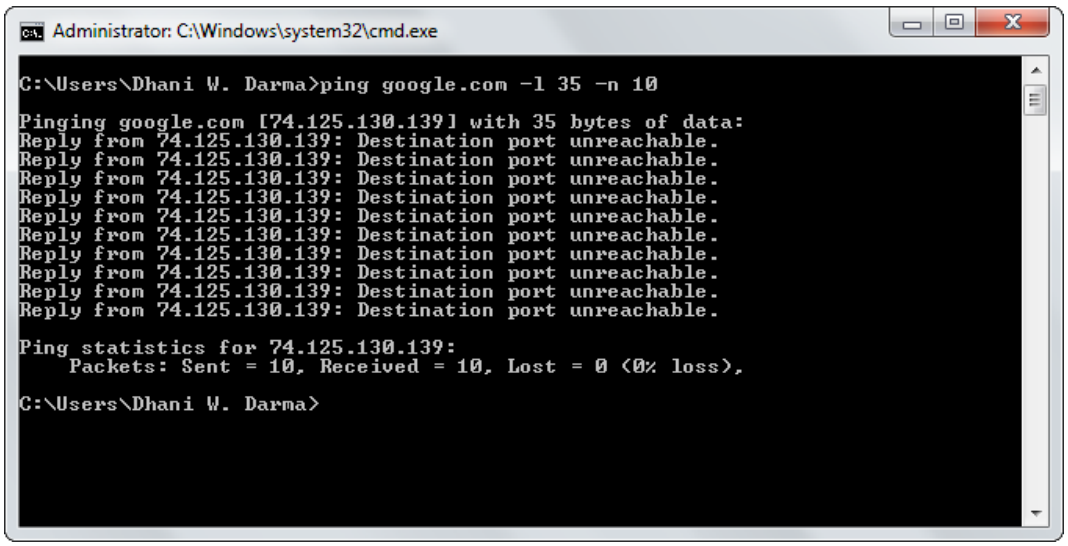
Pada kondisi normal, paket *ping* yang dikirimkan dapat diterima oleh *server* tujuan. Dalam hal ini *server* tujuan adalah www.google.com dengan *IP address* 74.125.200.139. Paket ping yang dikirimkan merupakan paket berukuran normal sehingga tidak termasuk dalam paket yang terindikasi sebagai sebuah serangan. Ping dengan kondisi normal dapat dilihat pada Gambar 4.3.



Gambar 4.3 Ping Kondisi Normal

Saat perintah *ping* dimodifikasi dengan mengubah besaran paket yang dikirim melebihi dari standar dan dengan jumlah request yang banyak, maka sistem IPS akan melakukan *drop* atau normalisasi terhadap paket tersebut. Dalam pengujian ini perintah ping yang digunakan adalah:

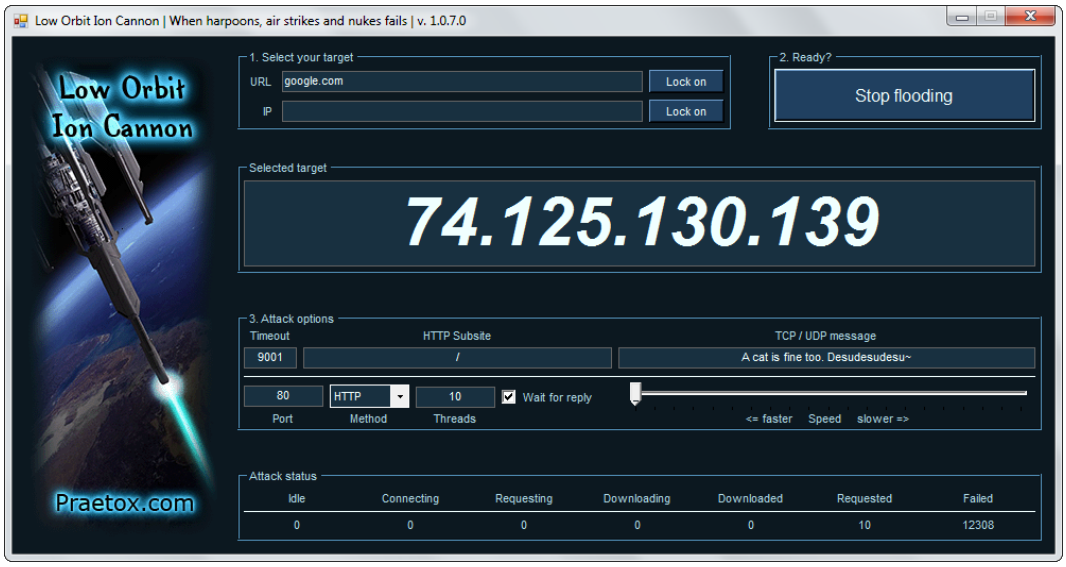
ping google.com –l 35 –n 10



Gambar 4.4 Ping Gagal Pada Komputer Penyerang

4.1.10 Hasil Pengujian dengan LOIC

Pengujian menggunakan perangkat lunak LOIC digunakan untuk melakukan *generate traffic* dengan jumlah yang banyak terhadap suatu server dengan target service HTTP server tersebut.



Gambar 4.5 Generate Traffic Dengan LOIC

4.1.11 Hasil Perbandingan Data

Sebelum sistem keamanan IPS diterapkan, *firewall* mencatat beberapa aktivitas pada jaringan yang tidak semestinya dengan *level critical* yang berasal dari server yang berada di jaringan DMZ, dalam kasus ini server menyerang salah satu host yang berada luar jaringan DMZ tanpa adanya tindakan penanganan keamanan. Apabila aktivitas tersebut terjadi terus-menerus, maka host yang menjadi target akan mengalami *crash/hang*. *Log* yang terekam pada aktivitas penyerangan tersebut, dapat di lihat pada Gambar 4.6 di bawah ini.

Date	Time	Module	Level	Description
02/02/2015	18:27:26	system	crit	Fragmented traffic! From 192.168.4.22:38656 to 27.111.185.144:443 (zone DMZ, int ethernet0/1). Occurred 150 times.
02/02/2015	18:27:13	system	crit	Fragmented traffic! From 192.168.4.22:9168 to 27.111.185.144:13368 (zone DMZ, int ethernet0/1). Occurred 151 times.
02/02/2015	18:26:56	system	crit	Fragmented traffic! From 192.168.4.249:25964 to 27.111.185.144:34523 (zone DMZ, int ethernet0/1). Occurred 151 times.
02/02/2015	18:26:34	system	crit	Fragmented traffic! From 192.168.4.22:14901 to 27.111.185.144:443 (zone DMZ, int ethernet0/1). Occurred 145 times.
02/02/2015	18:26:28	system	crit	Fragmented traffic! From 192.168.4.22:57600 to 27.111.185.144:443 (zone DMZ, int ethernet0/1). Occurred 145 times.
02/02/2015	18:26:19	system	crit	Fragmented traffic! From 192.168.4.249:60042 to 27.111.185.144:14495 (zone DMZ, int ethernet0/1). Occurred 106 times.
02/02/2015	18:25:12	system	crit	Fragmented traffic! From 192.168.4.22:6461 to 27.111.185.144:61771 (zone DMZ, int ethernet0/1). Occurred 121 times.
02/02/2015	18:25:03	system	crit	Fragmented traffic! From 192.168.4.22:54630 to 27.111.185.144:59485 (zone DMZ, int ethernet0/1). Occurred 127 times.
02/02/2015	18:23:36	system	crit	Fragmented traffic! From 192.168.4.249:24646 to 27.111.185.144:23829 (zone DMZ, int ethernet0/1). Occurred 157 times.
02/02/2015	18:23:29	system	crit	Fragmented traffic! From 192.168.4.22:63515 to 27.111.185.144:37043 (zone DMZ, int ethernet0/1). Occurred 157 times.
02/02/2015	18:23:21	system	crit	Fragmented traffic! From 192.168.4.22:37455 to 27.111.185.144:49138 (zone DMZ, int ethernet0/1). Occurred 102 times.
02/02/2015	18:23:14	system	crit	Fragmented traffic! From 192.168.4.22:32597 to 27.111.185.144:53161 (zone DMZ, int ethernet0/1). Occurred 110 times.

Gambar 4.6 *Log* pada *Firewall*

Setelah dilakukan penelitian dimulai dengan melakukan observasi di lapangan dan mengembangkan sistem keamanan, hingga pengujian dengan dua teknik yang berbeda, maka dihasilkan sebuah sistem keamanan yang diletakkan diantara *firewall* dan jaringan DMZ. Jika terjadi serangan dari dalam jaringan DMZ yang mengarah ke luar, sistem akan mendeteksi dan melakukan penanganan (tindak lanjut) untuk menghalau serangan tersebut. Sehingga serangan tersebut tidak sampai pada host tujuan yang menjadi target. *Log* yang ditunjukkan pada Gambar 4.7 merupakan hasil log sistem yang telah dirancang. Pada *log firewall*, tidak menampilkan serangan tersebut karena sebelumnya sudah di *drop* pada sistem keamanan IPS yang dirancang

ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(2-1442563)	[snort] Snort Alert [1:1104:1]	2015-02-02 18:31:35	192.168.4.22:10431	27.111.185.144:443	TCP
#1-(2-1436351)	[snort] Snort Alert [1:1104:1]	2015-02-02 18:31:30	192.168.4.22:10430	27.111.185.144:443	TCP
#2-(2-1430276)	[snort] Snort Alert [1:1104:1]	2015-02-02 18:31:26	192.168.4.22:10429	27.111.185.144:443	TCP
#3-(2-1400473)	[snort] Snort Alert [1:1104:1]	2015-02-02 18:31:07	192.168.4.22:10428	27.111.185.144:443	TCP
#4-(2-1400134)	[snort] Snort Alert [1:1104:1]	2015-02-02 18:31:07	192.168.4.22:10427	27.111.185.144:443	TCP
#5-(2-1390246)	[snort] Snort Alert [1:1104:1]	2015-02-02 18:31:00	192.168.4.22:10425	27.111.185.144:443	TCP
#6-(2-1390588)	[snort] Snort Alert [1:1104:1]	2015-02-02 18:31:00	192.168.4.22:10426	27.111.185.144:443	TCP
#7-(2-1380464)	[snort] Snort Alert [1:1104:1]	2015-02-02 18:30:54	192.168.4.22:10423	27.111.185.144:443	TCP
#8-(2-1380759)	[snort] Snort Alert [1:1104:1]	2015-02-02 18:30:54	192.168.4.22:10424	27.111.185.144:443	TCP

Gambar 4.7 *Log* pada Sistem Keamanan IPS

4.2 Pembahasan

Pada pengujian dengan perintah ping, pada Gambar 4.4 karena perintah ping di atas yang digunakan untuk pengujian telah melebihi standar besaran paket ping yang dimiliki oleh sistem operasi windows yaitu 32 bytes dengan request sebanyak 10 kali, maka sistem akan mendeteksinya sebagai sebuah pola awal dari serangan DoS sehingga sistem IPS akan melakukan *dropping* atau normalisasi terhadap paket tersebut dan mencatatnya. Gambar 4.4 menunjukan bahwa perintah ping gagal dilakukan pada komputer penyerang.

Hasil dari serangan tersebut juga dapat diamati pada sensor IPS yang telah dipasang BASE sebagai alat pelaporan serangan yang terjadi berdasarkan *log* yang dihasilkan dari sistem IPS. *Log* dapat dilihat pada Gambar 4.8.

☐	#28-(2-1340417)	[snort] PROTOCOL-ICMP PING	2014-12-02 18:30:27	192.168.4.212	74.125.130.139	ICMP
☐	#29-(2-1338899)	[snort] PROTOCOL-ICMP PING	2014-12-02 18:30:26	192.168.4.212	74.125.130.139	ICMP
☐	#30-(2-1337356)	[snort] PROTOCOL-ICMP PING	2014-12-02 18:30:25	192.168.4.212	74.125.130.139	ICMP
☐	#31-(2-1335811)	[snort] PROTOCOL-ICMP PING	2014-12-02 18:30:24	192.168.4.212	74.125.130.139	ICMP
☐	#32-(2-1334307)	[snort] PROTOCOL-ICMP PING	2014-12-02 18:30:23	192.168.4.212	74.125.130.139	ICMP
☐	#33-(2-1332767)	[snort] PROTOCOL-ICMP PING	2014-12-02 18:30:22	192.168.4.212	74.125.130.139	ICMP
☐	#34-(2-1331232)	[snort] PROTOCOL-ICMP PING	2014-12-02 18:30:21	192.168.4.212	74.125.130.139	ICMP
☐	#35-(2-1329708)	[snort] PROTOCOL-ICMP PING	2014-12-02 18:30:20	192.168.4.212	74.125.130.139	ICMP
☐	#36-(2-1328162)	[snort] PROTOCOL-ICMP PING	2014-12-02 18:30:19	192.168.4.212	74.125.130.139	ICMP
☐	#37-(2-1326632)	[snort] PROTOCOL-ICMP PING	2014-12-02 18:30:18	192.168.4.212	74.125.130.139	ICMP

Gambar 4.8 Pengamatan Ping Flood Pada BASE

Dari gambar 4.8 dapat dilihat BASE menampilkan informasi berupa ID serangan yang terjadi pada kolom pertama, kemudian pada kolom kedua menampilkan *signature/rules* pada snort yang cocok terhadap pola serangan *ping flood* yang dilakukan dengan nama *PROTOCOL-ICMP PING* dimana perintah ping yang dilakukan oleh penyerang tidak sesuai dengan standar yang terdapat pada *signature/rules* tersebut, selain itu juga ditampilkan waktu terjadinya insiden dan alamat ip sumber maupun alamat ip tujuan serta protokol yang digunakan.

Kemudian pada pengujian dengan LOIC, sistem IPS akan melakukan *drop* terhadap paket yang dikirimkan oleh LOIC dan pada aplikasi LOIC (Gambar 4.5)

dapat dilihat *request* yang berhasil masuk adalah 10 dan sebanyak 12308 *Failed*. Kemudian *log* yang dihasilkan sistem IPS ditampilkan pada BASE seperti yang ditunjukan pada Gambar 4.9.

ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(2-4381933)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16872	74.125.130.139:80	TCP
#1-(2-4381934)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16872	74.125.130.139:80	TCP
#2-(2-4381935)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16873	74.125.130.139:80	TCP
#3-(2-4381936)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16873	74.125.130.139:80	TCP
#4-(2-4381937)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16874	74.125.130.139:80	TCP
#5-(2-4381938)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16874	74.125.130.139:80	TCP
#6-(2-4381939)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16875	74.125.130.139:80	TCP
#7-(2-4381940)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16875	74.125.130.139:80	TCP
#8-(2-4381941)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16876	74.125.130.139:80	TCP
#9-(2-4381942)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16876	74.125.130.139:80	TCP
#10-(2-4381943)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16877	74.125.130.139:80	TCP
#11-(2-4381944)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16877	74.125.130.139:80	TCP
#12-(2-4381945)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16878	74.125.130.139:80	TCP
#13-(2-4381946)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16878	74.125.130.139:80	TCP
#14-(2-4381947)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16879	74.125.130.139:80	TCP
#15-(2-4381948)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16879	74.125.130.139:80	TCP
#16-(2-4381949)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16880	74.125.130.139:80	TCP
#17-(2-4381950)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16880	74.125.130.139:80	TCP
#18-(2-4381951)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16881	74.125.130.139:80	TCP
#19-(2-4381952)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:35	192.168.4.212:16881	74.125.130.139:80	TCP
#20-(2-4381698)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:08	192.168.4.212:16861	74.125.130.139:80	TCP
#21-(2-4381699)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:08	192.168.4.212:16861	74.125.130.139:80	TCP
#22-(2-4381700)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:08	192.168.4.212:16864	74.125.130.139:80	TCP
#23-(2-4381701)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:08	192.168.4.212:16864	74.125.130.139:80	TCP
#24-(2-4381702)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:08	192.168.4.212:16862	74.125.130.139:80	TCP
#25-(2-4381703)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:08	192.168.4.212:16862	74.125.130.139:80	TCP
#26-(2-4381704)	[snort] INDICATOR-SCAN Webtrends HTTP probe	2014-12-09 18:20:08	192.168.4.212:16863	74.125.130.139:80	TCP

Gambar 4.9 Pengamatan Traffic LOIC Pada BASE

Pada informasi yang ditampilkan gambar 4.9, untuk serangan yang dilakukan menggunakan aplikasi LOIC *signature/rules* snort mendeteksinya sebagai *INDICATOR-SCAN Webtrends HTTP Probe* dimana cara ini dilakukan untuk mengekploitasi kerentanan *service HTTP* dengan cara melakukan *request* dengan jumlah yang cukup banyak terhadap *service* tersebut yang dapat menyebabkan sebuah *service HTTP* dengan *webserver* di dalamnya menjadi *down*.

Dari kedua jenis pengujian tersebut, dapat disimpulkan bahwa sistem keamanan IPS tersebut telah mampu menghalau serangan yang dilakukan oleh komputer penyerang. Dari Gambar 4.4 dan 4.5 menandakan komputer penyerang gagal melakukan aksinya karena sebelum paket sampai pada server yang menjadi target, paket serangan terlebih dahulu melewati sistem keamanan IPS yang memeriksa apakah sebuah paket yang dikirim memiliki pola serangan atau tidak,

apabila paket yang dikirim ternyata memiliki pola yang sebelumnya telah ada pada *rules* snort sebagai IPS maka paket tersebut akan di *drop* dan tidak diteruskan ke *server* tujuan atau akan menormalisasi paket apabila dimungkinkan oleh sistem IPS.

Log yang dihasilkan oleh sistem IPS dapat dilihat melalui BASE sebagai alat pelaporan terhadap serangan yang sedang terjadi, akan tetapi BASE sebagai alat pelaporan masih memiliki kekurangan dalam menampilkan keterangan apakah suatu paket itu di *drop* atau hanya sebuah *alert* dari paket yang berpotensi sebagai sebuah serangan. Karena pada tampilan BASE hanya menampilkan ID serangan, informasi *signature/rules* yang cocok dengan serangan yang terjadi, waktu terjadinya serangan, ip sumber dan ip tujuan serta protokol yang digunakan, sehingga informasi yang disajikan masih kurang memadai.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan dari analisa pada bab-bab sebelumnya dan teori yang ada, maka dapat ditarik kesimpulan:

1. Keamanan jaringan komputer menggunakan *Intrusion Prevention System* dengan Snort telah dapat menghalau setiap serangan yang dilakukan berupa *ping flood* dan *DoS* menggunakan LOIC dengan *target service HTTP*.
2. Sistem keamanan IPS dapat menampilkan *log event* serangan yang terjadi dalam bentuk tampilan berbasis *web* dengan menggunakan BASE.

5.2 Saran

Saran-saran yang dapat peneliti berikan berdasarkan analisa dan kesimpulan yaitu:

1. Sistem keamanan IPS ini masih memiliki kekurangan ketika menampilkan *log* peringatan yang dihasilkan oleh snort pada BASE yaitu laporan yang akan ditampilkan memiliki jeda yang cukup lama dengan serangan yang sedang terjadi. Hal ini kemungkinan terjadi karena spesifikasi perangkat keras yang digunakan kurang memadai. Untuk pengembangan selanjutnya sebaiknya perangkat yang digunakan memiliki spesifikasi yang lebih tinggi dari yang ada saat ini.
2. BASE yang dijadikan sebagai alat pelaporan dan analisis tidak dapat menampilkan informasi secara details apakah paket yang tercatat tersebut telah di *drop* atau hanya sebuah peringatan dari paket yang berpotensi

sebagai sebuah serangan. Sehingga perlu alternatif alat pelaporan yang dapat menampilkan informasi secara lebih details.

3. *Rules* memiliki peran yang sangat penting, agar sistem keamanan IPS ini dapat mendeteksi dan melakukan *drop* terhadap serangan terbaru, maka sebaiknya *rules* diperbarui secara berkala menggunakan *pulledpork*. Bila diperlukan dapat berlangganan *rules* berbayar yang disediakan oleh snort.

DAFTAR PUSTAKA

Arius, Dony. 2006. *Computer Security*. Yogyakarta: ANDI.

Brennan, Michael P.. 2002. *Using Snort For a Distributed Intrusion Detection System*. SANS Institute.

Cisco. 2014. *SNORT Users Manual 2.9.7*. [terhubung berkala]. http://manual.snort.org/ [24 Agustus 2014]

Cloara, Jeremy, dkk.. 2008. *CCNA Examp Prep, Second Edition*. USA: Pearson Education, Inc..

Dewannanta, Didha. 2012. *Tujuan, Risiko dan Ancaman pada Keamanan Jaringan Komputer*. IlmuKomputer.com

Easttom, Chuck. 2012. *Computer Security Fundamentals*. Indianapolis: Pearson.

Gondohanindijo, Jutono. 2012. *IPS (Intrusion Prevention System) Untuk Mencegah Tindak Penyusupan/Intrusi*. Majalah Ilmiah INFORMATIKA Vol. 3 No. 3.

Madcoms. 2009. *Membangun Sistem Jaringan Komputer*. Yogyakarta: ANDI.

Moya, Miguel A. Calvo. 2008. *Analysis And Evaluation Of The Snort And Bro Network Intrusion Detection Systems*. Madrid. Comillas Pontifical University

Nugroho, Arianto C. 2005. *Keamanan Jaringan Komputer*. Depok: Makalah MTI UI.

Scarfone, Karen and Mell, Peter. 2007. *Guide to Intrusion Detection and Prevention Systems (IDPS)*. Gaithersburg: NIST.

Sourcefire. *Snort IDS/IPS + Rule Writing Technology*. Sourcefire Education Courses

Sugiyono. 2013. *Metode Penelitian Kuantitatif Kualitatif dan R&D*. Bandung: Alfabeta.

Tanenbaum, Andrew S. 2003. *Computer Network, Fourth Edition*. New Jersey: Prentice Hall.

Tim Penyusun. 2012. *Buku Pedoman Skripsi/Komprehensif/Karya Inovatif*. Jakarta: Universitas Negeri Jakarta.

Lampiran 1. Log Firewall Universitas Negeri Jakarta 28 April 2014

Date	Time	Level	Description
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:60042 to 39.50.197.165:14495 (zone DMZ, int ethernet0/1). Occurred 23 times.
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:6461 to 39.50.197.165:61771 (zone DMZ, int ethernet0/1). Occurred 23 times.
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:30482 to 39.50.197.165:26532 (zone DMZ, int ethernet0/1). Occurred 18 times.
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:25360 to 39.50.197.165:29816 (zone DMZ, int ethernet0/1). Occurred 18 times.
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:42412 to 39.50.197.165:50715 (zone DMZ, int ethernet0/1). Occurred 18 times.
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:54630 to 39.50.197.165:59485 (zone DMZ, int ethernet0/1). Occurred 25 times.
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:24646 to 39.50.197.165:23829 (zone DMZ, int ethernet0/1). Occurred 25 times.
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:63515 to 39.50.197.165:37043 (zone DMZ, int ethernet0/1). Occurred 25 times.
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:57392 to 39.50.197.165:24485 (zone DMZ, int ethernet0/1). Occurred 18 times.
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:57522 to 39.50.197.165:41117 (zone DMZ, int ethernet0/1). Occurred 18 times.
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:46930 to 39.50.197.165:42980 (zone DMZ, int ethernet0/1). Occurred 19 times.
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:41808 to 39.50.197.165:46264 (zone DMZ, int ethernet0/1). Occurred 19 times.
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:58860 to 39.50.197.165:1627 (zone DMZ, int ethernet0/1). Occurred 19 times.
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:10954 to 39.50.197.165:30943 (zone DMZ, int ethernet0/1). Occurred 26 times.
28/04/2014	11:06:17	emer	Teardrop attack! From 192.168.4.249:22909 to 39.50.197.165:12683 (zone DMZ, int ethernet0/1). Occurred 26 times.

28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:5542 to 39.50.197.165:10397 (zone DMZ, int ethernet0/1). Occurred 26 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:41094 to 39.50.197.165:40277 (zone DMZ, int ethernet0/1). Occurred 26 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:14427 to 39.50.197.165:53491 (zone DMZ, int ethernet0/1). Occurred 26 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:8304 to 39.50.197.165:40933 (zone DMZ, int ethernet0/1). Occurred 19 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:8434 to 39.50.197.165:57565 (zone DMZ, int ethernet0/1). Occurred 19 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:63378 to 39.50.197.165:59172 (zone DMZ, int ethernet0/1). Occurred 19 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:58256 to 39.50.197.165:62712 (zone DMZ, int ethernet0/1). Occurred 19 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:9516 to 39.50.197.165:18075 (zone DMZ, int ethernet0/1). Occurred 19 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:27146 to 39.50.197.165:47135 (zone DMZ, int ethernet0/1). Occurred 25 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:39357 to 39.50.197.165:29131 (zone DMZ, int ethernet0/1). Occurred 25 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:21990 to 39.50.197.165:26845 (zone DMZ, int ethernet0/1). Occurred 25 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:57542 to 39.50.197.165:56725 (zone DMZ, int ethernet0/1). Occurred 25 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:30875 to 39.50.197.165:4147 (zone DMZ, int ethernet0/1). Occurred 25 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:24752 to 39.50.197.165:57125 (zone DMZ, int ethernet0/1). Occurred 19 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:24626 to 39.50.197.165:8221 (zone DMZ, int ethernet0/1). Occurred 19 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:14290 to 39.50.197.165:10084 (zone DMZ, int ethernet0/1). Occurred 19 times.

28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:9168 to 39.50.197.165:13368 (zone DMZ, int ethernet0/1). Occurred 19 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:25964 to 39.50.197.165:34523 (zone DMZ, int ethernet0/1). Occurred 19 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:43594 to 39.50.197.165:63583 (zone DMZ, int ethernet0/1). Occurred 23 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:55805 to 39.50.197.165:45323 (zone DMZ, int ethernet0/1). Occurred 23 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:38182 to 39.50.197.165:43037 (zone DMZ, int ethernet0/1). Occurred 23 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:8198 to 39.50.197.165:7637 (zone DMZ, int ethernet0/1). Occurred 23 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:47323 to 39.50.197.165:20595 (zone DMZ, int ethernet0/1). Occurred 23 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:41200 to 39.50.197.165:8037 (zone DMZ, int ethernet0/1). Occurred 19 times.
28/04 /2014	11:06: 17	emer	Teardrop attack! From 192.168.4.249:41074 to 39.50.197.165:24669 (zone DMZ, int ethernet0/1). Occurred 19 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:38182 to 39.50.197.165:43037 (zone DMZ, int ethernet0/1). Occurred 33 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:8198 to 39.50.197.165:7637 (zone DMZ, int ethernet0/1). Occurred 33 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:47323 to 39.50.197.165:20595 (zone DMZ, int ethernet0/1). Occurred 33 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:60042 to 39.50.197.165:14495 (zone DMZ, int ethernet0/1). Occurred 34 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:6461 to 39.50.197.165:61771 (zone DMZ, int ethernet0/1). Occurred 34 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:42212 to 39.50.197.165:18793 (zone DMZ, int ethernet0/1). Occurred 21 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:23788 to 39.50.197.165:13298 (zone DMZ, int ethernet0/1). Occurred 21 times.

28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:41200 to 39.50.197.165:8037 (zone DMZ, int ethernet0/1). Occurred 34 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:41074 to 39.50.197.165:24669 (zone DMZ, int ethernet0/1). Occurred 34 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:30482 to 39.50.197.165:26532 (zone DMZ, int ethernet0/1). Occurred 34 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:25360 to 39.50.197.165:29816 (zone DMZ, int ethernet0/1). Occurred 34 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:42412 to 39.50.197.165:50715 (zone DMZ, int ethernet0/1). Occurred 34 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:53903 to 39.50.197.165:65330 (zone DMZ, int ethernet0/1). Occurred 23 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:49045 to 39.50.197.165:4073 (zone DMZ, int ethernet0/1). Occurred 22 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:23473 to 39.50.197.165:32166 (zone DMZ, int ethernet0/1). Occurred 22 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:58404 to 39.50.197.165:35241 (zone DMZ, int ethernet0/1). Occurred 22 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:39980 to 39.50.197.165:29490 (zone DMZ, int ethernet0/1). Occurred 22 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:54630 to 39.50.197.165:59485 (zone DMZ, int ethernet0/1). Occurred 36 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:24646 to 39.50.197.165:23829 (zone DMZ, int ethernet0/1). Occurred 36 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:63515 to 39.50.197.165:37043 (zone DMZ, int ethernet0/1). Occurred 36 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:57392 to 39.50.197.165:24485 (zone DMZ, int ethernet0/1). Occurred 33 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:57522 to 39.50.197.165:41117 (zone DMZ, int ethernet0/1). Occurred 33 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:46930 to 39.50.197.165:42980 (zone DMZ, int ethernet0/1). Occurred 34 times.

28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:41808 to 39.50.197.165:46264 (zone DMZ, int ethernet0/1). Occurred 34 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:58860 to 39.50.197.165:1627 (zone DMZ, int ethernet0/1). Occurred 34 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:4815 to 39.50.197.165:16242 (zone DMZ, int ethernet0/1). Occurred 21 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:65493 to 39.50.197.165:20265 (zone DMZ, int ethernet0/1). Occurred 21 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:39921 to 39.50.197.165:48614 (zone DMZ, int ethernet0/1). Occurred 22 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:10954 to 39.50.197.165:30943 (zone DMZ, int ethernet0/1). Occurred 35 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:9316 to 39.50.197.165:51689 (zone DMZ, int ethernet0/1). Occurred 22 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:22909 to 39.50.197.165:12683 (zone DMZ, int ethernet0/1). Occurred 35 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:56428 to 39.50.197.165:45938 (zone DMZ, int ethernet0/1). Occurred 22 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:5542 to 39.50.197.165:10397 (zone DMZ, int ethernet0/1). Occurred 35 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:41094 to 39.50.197.165:40277 (zone DMZ, int ethernet0/1). Occurred 35 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:14427 to 39.50.197.165:53491 (zone DMZ, int ethernet0/1). Occurred 35 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:8304 to 39.50.197.165:40933 (zone DMZ, int ethernet0/1). Occurred 35 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:8434 to 39.50.197.165:57565 (zone DMZ, int ethernet0/1). Occurred 35 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:63378 to 39.50.197.165:59172 (zone DMZ, int ethernet0/1). Occurred 35 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:58256 to 39.50.197.165:62712 (zone DMZ, int ethernet0/1). Occurred 35 times.

28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:9516 to 39.50.197.165:18075 (zone DMZ, int ethernet0/1). Occurred 35 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:21007 to 39.50.197.165:32690 (zone DMZ, int ethernet0/1). Occurred 24 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:16149 to 39.50.197.165:36713 (zone DMZ, int ethernet0/1). Occurred 24 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:27146 to 39.50.197.165:47135 (zone DMZ, int ethernet0/1). Occurred 34 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:56113 to 39.50.197.165:64806 (zone DMZ, int ethernet0/1). Occurred 24 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:39357 to 39.50.197.165:29131 (zone DMZ, int ethernet0/1). Occurred 33 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:25764 to 39.50.197.165:2345 (zone DMZ, int ethernet0/1). Occurred 24 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:21990 to 39.50.197.165:26845 (zone DMZ, int ethernet0/1). Occurred 33 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:7340 to 39.50.197.165:62386 (zone DMZ, int ethernet0/1). Occurred 24 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:57542 to 39.50.197.165:56725 (zone DMZ, int ethernet0/1). Occurred 32 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:30875 to 39.50.197.165:4147 (zone DMZ, int ethernet0/1). Occurred 32 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:24752 to 39.50.197.165:57125 (zone DMZ, int ethernet0/1). Occurred 36 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:24626 to 39.50.197.165:8221 (zone DMZ, int ethernet0/1). Occurred 36 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:14290 to 39.50.197.165:10084 (zone DMZ, int ethernet0/1). Occurred 36 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:9168 to 39.50.197.165:13368 (zone DMZ, int ethernet0/1). Occurred 36 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:25964 to 39.50.197.165:34523 (zone DMZ, int ethernet0/1). Occurred 36 times.

28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:37455 to 39.50.197.165:49138 (zone DMZ, int ethernet0/1). Occurred 22 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:43594 to 39.50.197.165:63583 (zone DMZ, int ethernet0/1). Occurred 34 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:32597 to 39.50.197.165:53161 (zone DMZ, int ethernet0/1). Occurred 22 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:55805 to 39.50.197.165:45323 (zone DMZ, int ethernet0/1). Occurred 34 times.
28/04 /2014	11:06: 15	emer	Teardrop attack! From 192.168.4.249:7025 to 39.50.197.165:15718 (zone DMZ, int ethernet0/1). Occurred 22 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:24646 to 39.50.197.165:23829 (zone DMZ, int ethernet0/1). Occurred 3 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:41808 to 39.50.197.165:46264 (zone DMZ, int ethernet0/1). Occurred 3 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:9168 to 39.50.197.165:13368 (zone DMZ, int ethernet0/1). Occurred 3 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:25764 to 39.50.197.165:2345 (zone DMZ, int ethernet0/1). Occurred 3 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:58404 to 39.50.197.165:35241 (zone DMZ, int ethernet0/1). Occurred 2 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:57542 to 39.50.197.165:56725 (zone DMZ, int ethernet0/1). Occurred 3 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:30875 to 39.50.197.165:4147 (zone DMZ, int ethernet0/1). Occurred 1 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:55805 to 39.50.197.165:45323 (zone DMZ, int ethernet0/1). Occurred 24 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:38182 to 39.50.197.165:43037 (zone DMZ, int ethernet0/1). Occurred 23 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:57522 to 39.50.197.165:41117 (zone DMZ, int ethernet0/1). Occurred 23 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:46930 to 39.50.197.165:42980 (zone DMZ, int ethernet0/1). Occurred 23 times.

28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:65493 to 39.50.197.165:20265 (zone DMZ, int ethernet0/1). Occurred 33 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:39921 to 39.50.197.165:48614 (zone DMZ, int ethernet0/1). Occurred 29 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:43594 to 39.50.197.165:63583 (zone DMZ, int ethernet0/1). Occurred 46 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:6461 to 39.50.197.165:61771 (zone DMZ, int ethernet0/1). Occurred 26 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:54630 to 39.50.197.165:59485 (zone DMZ, int ethernet0/1). Occurred 23 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:41200 to 39.50.197.165:8037 (zone DMZ, int ethernet0/1). Occurred 48 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:41074 to 39.50.197.165:24669 (zone DMZ, int ethernet0/1). Occurred 32 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:30482 to 39.50.197.165:26532 (zone DMZ, int ethernet0/1). Occurred 29 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:49045 to 39.50.197.165:4073 (zone DMZ, int ethernet0/1). Occurred 29 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:23473 to 39.50.197.165:32166 (zone DMZ, int ethernet0/1). Occurred 21 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:39357 to 39.50.197.165:29131 (zone DMZ, int ethernet0/1). Occurred 29 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:21990 to 39.50.197.165:26845 (zone DMZ, int ethernet0/1). Occurred 28 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:4815 to 39.50.197.165:16242 (zone DMZ, int ethernet0/1). Occurred 41 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:8304 to 39.50.197.165:40933 (zone DMZ, int ethernet0/1). Occurred 34 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:8434 to 39.50.197.165:57565 (zone DMZ, int ethernet0/1). Occurred 28 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:63378 to 39.50.197.165:59172 (zone DMZ, int ethernet0/1). Occurred 25 times.

28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:21007 to 39.50.197.165:32690 (zone DMZ, int ethernet0/1). Occurred 39 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:16149 to 39.50.197.165:36713 (zone DMZ, int ethernet0/1). Occurred 29 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:56113 to 39.50.197.165:64806 (zone DMZ, int ethernet0/1). Occurred 22 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:24752 to 39.50.197.165:57125 (zone DMZ, int ethernet0/1). Occurred 36 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:24626 to 39.50.197.165:8221 (zone DMZ, int ethernet0/1). Occurred 28 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:14290 to 39.50.197.165:10084 (zone DMZ, int ethernet0/1). Occurred 26 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:60042 to 39.50.197.165:14495 (zone DMZ, int ethernet0/1). Occurred 45 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:37455 to 39.50.197.165:49138 (zone DMZ, int ethernet0/1). Occurred 46 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:32597 to 39.50.197.165:53161 (zone DMZ, int ethernet0/1). Occurred 33 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:7025 to 39.50.197.165:15718 (zone DMZ, int ethernet0/1). Occurred 25 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:10954 to 39.50.197.165:30943 (zone DMZ, int ethernet0/1). Occurred 36 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:22909 to 39.50.197.165:12683 (zone DMZ, int ethernet0/1). Occurred 29 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:5542 to 39.50.197.165:10397 (zone DMZ, int ethernet0/1). Occurred 25 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:53903 to 39.50.197.165:65330 (zone DMZ, int ethernet0/1). Occurred 44 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:57392 to 39.50.197.165:24485 (zone DMZ, int ethernet0/1). Occurred 42 times.
28/04 /2014	11:06: 13	emer	Teardrop attack! From 192.168.4.249:27146 to 39.50.197.165:47135 (zone DMZ, int ethernet0/1). Occurred 36 times.

28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:30482 to 39.50.197.165:26532 (zone DMZ, int ethernet0/1). Occurred 1 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:38182 to 39.50.197.165:43037 (zone DMZ, int ethernet0/1). Occurred 1 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:63378 to 39.50.197.165:59172 (zone DMZ, int ethernet0/1). Occurred 1 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:54630 to 39.50.197.165:59485 (zone DMZ, int ethernet0/1). Occurred 1 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:21990 to 39.50.197.165:26845 (zone DMZ, int ethernet0/1). Occurred 1 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:56113 to 39.50.197.165:64806 (zone DMZ, int ethernet0/1). Occurred 3 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:25764 to 39.50.197.165:2345 (zone DMZ, int ethernet0/1). Occurred 2 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:41074 to 39.50.197.165:24669 (zone DMZ, int ethernet0/1). Occurred 3 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:27146 to 39.50.197.165:47135 (zone DMZ, int ethernet0/1). Occurred 5 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:39357 to 39.50.197.165:29131 (zone DMZ, int ethernet0/1). Occurred 5 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:4815 to 39.50.197.165:16242 (zone DMZ, int ethernet0/1). Occurred 4 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:65493 to 39.50.197.165:20265 (zone DMZ, int ethernet0/1). Occurred 3 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:16149 to 39.50.197.165:36713 (zone DMZ, int ethernet0/1). Occurred 4 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:10954 to 39.50.197.165:30943 (zone DMZ, int ethernet0/1). Occurred 5 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:22909 to 39.50.197.165:12683 (zone DMZ, int ethernet0/1). Occurred 4 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:37455 to 39.50.197.165:49138 (zone DMZ, int ethernet0/1). Occurred 4 times.

28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:32597 to 39.50.197.165:53161 (zone DMZ, int ethernet0/1). Occurred 2 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:41200 to 39.50.197.165:8037 (zone DMZ, int ethernet0/1). Occurred 7 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:53903 to 39.50.197.165:65330 (zone DMZ, int ethernet0/1). Occurred 6 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:49045 to 39.50.197.165:4073 (zone DMZ, int ethernet0/1). Occurred 4 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:57392 to 39.50.197.165:24485 (zone DMZ, int ethernet0/1). Occurred 5 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:57522 to 39.50.197.165:41117 (zone DMZ, int ethernet0/1). Occurred 4 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:43594 to 39.50.197.165:63583 (zone DMZ, int ethernet0/1). Occurred 6 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:55805 to 39.50.197.165:45323 (zone DMZ, int ethernet0/1). Occurred 4 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:8304 to 39.50.197.165:40933 (zone DMZ, int ethernet0/1). Occurred 6 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:8434 to 39.50.197.165:57565 (zone DMZ, int ethernet0/1). Occurred 5 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:48695 to 39.50.197.165:4935 (zone DMZ, int ethernet0/1). Occurred 42 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:38656 to 39.50.197.165:31367 (zone DMZ, int ethernet0/1). Occurred 40 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:14901 to 39.50.197.165:41565 (zone DMZ, int ethernet0/1). Occurred 41 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:60042 to 39.50.197.165:14495 (zone DMZ, int ethernet0/1). Occurred 6 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:6461 to 39.50.197.165:61771 (zone DMZ, int ethernet0/1). Occurred 5 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:41358 to 39.50.197.165:41015 (zone DMZ, int ethernet0/1). Occurred 41 times.

28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:21007 to 39.50.197.165:32690 (zone DMZ, int ethernet0/1). Occurred 6 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:24752 to 39.50.197.165:57125 (zone DMZ, int ethernet0/1). Occurred 5 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.249:24626 to 39.50.197.165:8221 (zone DMZ, int ethernet0/1). Occurred 2 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:6935 to 39.50.197.165:50101 (zone DMZ, int ethernet0/1). Occurred 40 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:57600 to 39.50.197.165:37856 (zone DMZ, int ethernet0/1). Occurred 43 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:11189 to 39.50.197.165:41116 (zone DMZ, int ethernet0/1). Occurred 42 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:25514 to 39.50.197.165:29530 (zone DMZ, int ethernet0/1). Occurred 41 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:60117 to 39.50.197.165:17469 (zone DMZ, int ethernet0/1). Occurred 41 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:65143 to 39.50.197.165:21383 (zone DMZ, int ethernet0/1). Occurred 42 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:57806 to 39.50.197.165:57463 (zone DMZ, int ethernet0/1). Occurred 39 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:23383 to 39.50.197.165:1013 (zone DMZ, int ethernet0/1). Occurred 46 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:55104 to 39.50.197.165:47815 (zone DMZ, int ethernet0/1). Occurred 47 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:8928 to 39.50.197.165:61247 (zone DMZ, int ethernet0/1). Occurred 47 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:31349 to 39.50.197.165:58013 (zone DMZ, int ethernet0/1). Occurred 48 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:8512 to 39.50.197.165:54048 (zone DMZ, int ethernet0/1). Occurred 46 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:27637 to 39.50.197.165:57564 (zone DMZ, int ethernet0/1). Occurred 40 times.

28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:41962 to 39.50.197.165:45978 (zone DMZ, int ethernet0/1). Occurred 40 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:10773 to 39.50.197.165:33917 (zone DMZ, int ethernet0/1). Occurred 41 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:16055 to 39.50.197.165:37831 (zone DMZ, int ethernet0/1). Occurred 41 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:8462 to 39.50.197.165:8375 (zone DMZ, int ethernet0/1). Occurred 40 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:39831 to 39.50.197.165:17205 (zone DMZ, int ethernet0/1). Occurred 45 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:6016 to 39.50.197.165:64007 (zone DMZ, int ethernet0/1). Occurred 44 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:47797 to 39.50.197.165:8925 (zone DMZ, int ethernet0/1). Occurred 41 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:24960 to 39.50.197.165:4960 (zone DMZ, int ethernet0/1). Occurred 41 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:43829 to 39.50.197.165:8220 (zone DMZ, int ethernet0/1). Occurred 41 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:58154 to 39.50.197.165:62426 (zone DMZ, int ethernet0/1). Occurred 42 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:27221 to 39.50.197.165:50365 (zone DMZ, int ethernet0/1). Occurred 44 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:32503 to 39.50.197.165:54023 (zone DMZ, int ethernet0/1). Occurred 43 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:24910 to 39.50.197.165:24823 (zone DMZ, int ethernet0/1). Occurred 41 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:56279 to 39.50.197.165:33653 (zone DMZ, int ethernet0/1). Occurred 45 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:22464 to 39.50.197.165:14919 (zone DMZ, int ethernet0/1). Occurred 43 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:41568 to 39.50.197.165:28607 (zone DMZ, int ethernet0/1). Occurred 44 times.

28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:64245 to 39.50.197.165:25117 (zone DMZ, int ethernet0/1). Occurred 43 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:41408 to 39.50.197.165:21408 (zone DMZ, int ethernet0/1). Occurred 43 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:60277 to 39.50.197.165:24668 (zone DMZ, int ethernet0/1). Occurred 46 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:9066 to 39.50.197.165:13082 (zone DMZ, int ethernet0/1). Occurred 44 times.
28/04 /2014	11:06: 12	emer	Teardrop attack! From 192.168.4.22:43669 to 39.50.197.165:1277 (zone DMZ, int ethernet0/1). Occurred 43 times.
28/04 /2014	11:06: 11	emer	Teardrop attack! From 192.168.4.249:57542 to 39.50.197.165:56725 (zone DMZ, int ethernet0/1). Occurred 1 times.
28/04 /2014	11:06: 11	emer	Teardrop attack! From 192.168.4.249:58404 to 39.50.197.165:35241 (zone DMZ, int ethernet0/1). Occurred 1 times.
28/04 /2014	11:06: 11	emer	Teardrop attack! From 192.168.4.249:9316 to 39.50.197.165:51689 (zone DMZ, int ethernet0/1). Occurred 2 times.
28/04 /2014	11:06: 11	emer	Teardrop attack! From 192.168.4.249:39357 to 39.50.197.165:29131 (zone DMZ, int ethernet0/1). Occurred 33 times.
28/04 /2014	11:06: 11	emer	Teardrop attack! From 192.168.4.249:21990 to 39.50.197.165:26845 (zone DMZ, int ethernet0/1). Occurred 33 times.
28/04 /2014	11:06: 11	emer	Teardrop attack! From 192.168.4.249:8434 to 39.50.197.165:57565 (zone DMZ, int ethernet0/1). Occurred 20 times.
28/04 /2014	11:06: 11	emer	Teardrop attack! From 192.168.4.249:63378 to 39.50.197.165:59172 (zone DMZ, int ethernet0/1). Occurred 20 times.
28/04 /2014	11:06: 11	emer	Teardrop attack! From 192.168.4.249:6461 to 39.50.197.165:61771 (zone DMZ, int ethernet0/1). Occurred 30 times.
28/04 /2014	11:06: 11	emer	Teardrop attack! From 192.168.4.249:54630 to 39.50.197.165:59485 (zone DMZ, int ethernet0/1). Occurred 30 times.
28/04 /2014	11:06: 11	emer	Teardrop attack! From 192.168.4.249:41074 to 39.50.197.165:24669 (zone DMZ, int ethernet0/1). Occurred 22 times.
28/04 /2014	11:06: 11	emer	Teardrop attack! From 192.168.4.249:30482 to 39.50.197.165:26532 (zone DMZ, int ethernet0/1). Occurred 22 times.

28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.249:24626 to 39.50.197.165:8221 (zone DMZ, int ethernet0/1). Occurred 3 times.
28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.249:14290 to 39.50.197.165:10084 (zone DMZ, int ethernet0/1). Occurred 1 times.
28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.249:60042 to 39.50.197.165:14495 (zone DMZ, int ethernet0/1). Occurred 5 times.
28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.249:6461 to 39.50.197.165:61771 (zone DMZ, int ethernet0/1). Occurred 5 times.
28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.249:53903 to 39.50.197.165:65330 (zone DMZ, int ethernet0/1). Occurred 4 times.
28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.249:49045 to 39.50.197.165:4073 (zone DMZ, int ethernet0/1). Occurred 4 times.
28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.22:23383 to 39.50.197.165:1013 (zone DMZ, int ethernet0/1). Occurred 44 times.
28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.22:55104 to 39.50.197.165:47815 (zone DMZ, int ethernet0/1). Occurred 44 times.
28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.22:8928 to 39.50.197.165:61247 (zone DMZ, int ethernet0/1). Occurred 46 times.
28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.22:31349 to 39.50.197.165:58013 (zone DMZ, int ethernet0/1). Occurred 41 times.
28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.22:8512 to 39.50.197.165:54048 (zone DMZ, int ethernet0/1). Occurred 41 times.
28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.22:27637 to 39.50.197.165:57564 (zone DMZ, int ethernet0/1). Occurred 46 times.
28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.22:41962 to 39.50.197.165:45978 (zone DMZ, int ethernet0/1). Occurred 45 times.
28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.22:10773 to 39.50.197.165:33917 (zone DMZ, int ethernet0/1). Occurred 43 times.
28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.22:16055 to 39.50.197.165:37831 (zone DMZ, int ethernet0/1). Occurred 43 times.
28/04 /2014	11:06: 10	emer	Teardrop attack! From 192.168.4.22:8462 to 39.50.197.165:8375 (zone DMZ, int ethernet0/1). Occurred 44 times.

Lampiran 2. File Konfigurasi Snort (snort.conf)

```
#--------------------------------------------------
#   VRT Rule Packages Snort.conf
#
#   For more information visit us at:
#     http://www.snort.org                   Snort Website
#     http://vrt-blog.snort.org/    Sourcefire VRT Blog
#
#             Mailing   list   Contact:                snort-
sigs@lists.sourceforge.net
#     False Positive reports:    fp@sourcefire.com
#     Snort bugs:                bugs@snort.org
#
#     Compatible with Snort Versions:
#     VERSIONS : 2.9.6.2
#
#     Snort build options:
#          OPTIONS  :  --enable-gre  --enable-mpls  --enable-
targetbased  --enable-ppm  --enable-perfprofiling  --enable-
zlib --enable-active-response --enable-normalizer --enable-
reload --enable-react --enable-flexresp3
#
#     Additional information:
#       This configuration file enables active response, to
run snort in
#     test mode -T you are required to supply an interface -
i <interface>
#         or  test  mode  will  fail  to  fully  validate  the
configuration and
#     exit with a FATAL error
#--------------------------------------------------

######################################################
# This file contains a sample snort configuration.
# You should take the following steps to create your own
custom configuration:
#
#  1) Set the network variables.
#  2) Configure the decoder
#  3) Configure the base detection engine
#  4) Configure dynamic loaded libraries
#  5) Configure preprocessors
#  6) Configure output plugins
#  7) Customize your rule set
#  8) Customize preprocessor and decoder rule set
#  9) Customize shared object rule set
######################################################

######################################################
# Step #1: Set the network variables.  For more information,
see README.variables
######################################################
```

```
# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.4.0/24

# Set up the external network addresses. Leave as "any" in
most situations
ipvar EXTERNAL_NET any

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET

# List of web servers on your network
ipvar HTTP_SERVERS $HOME_NET

# List of sql servers on your network
ipvar SQL_SERVERS $HOME_NET

# List of telnet servers on your network
ipvar TELNET_SERVERS $HOME_NET

# List of ssh servers on your network
ipvar SSH_SERVERS $HOME_NET

# List of ftp servers on your network
ipvar FTP_SERVERS $HOME_NET

# List of sip servers on your network
ipvar SIP_SERVERS $HOME_NET

# List of ports you run web servers on portvar HTTP_PORTS
[36,80,81,82,83,84,85,86,87,88,89,90,311,383,555,591,593,631
,801,808,818,901,972,1158,1220,1414,1533,1741,1830,1942,2231
,2301,2381,2578,2809,2980,3029,3037,3057,3128,3443,3702,4000
,4343,4848,5000,5117,5250,5600,6080,6173,6988,7000,7001,7071
,7144,7145,7510,7770,7777,7778,7779,8000,8008,8014,8028,8080
,8081,8082,8085,8088,8090,8118,8123,8180,8181,8222,8243,8280
,8300,8333,8344,8500,8509,8800,8888,8899,8983,9000,9060,9080
,9090,9091,9111,9290,9443,9999,10000,11371,12601,13014,15489
,29991,33300,34412,34443,34444,41080,44449,50000,50002,51423
,53331,55252,55555,56712]

# List of ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80

# List of ports you might see oracle attacks on
portvar ORACLE_PORTS 1024:

# List of ports you want to look for SSH connections on:
portvar SSH_PORTS 22
```

```
# List of ports you run ftp servers on
portvar FTP_PORTS [21,2100,3535]

# List of ports you run SIP servers on
portvar SIP_PORTS [5060,5061,5600]

# List of file data ports for file inspection
portvar FILE_DATA_PORTS [$HTTP_PORTS,110,143]

# List of GTP ports for GTP preprocessor
portvar GTP_PORTS [2123,2152,3386]

# other variables, these should not be modified ipvar
AIM_SERVERS
[64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,6
4.12.200.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,2
05.188.9.0/24,205.188.153.0/24,205.188.179.0/24,205.188.248.
0/24]

# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an
absolute path,
# such as:  c:\snort\rules
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules

# If you are using reputation preprocessor set these
var WHITE_LIST_PATH ../rules
var BLACK_LIST_PATH ../rules

##################################################
# Step #2: Configure the decoder.  For more information, see
README.decode
##################################################

# Stop generic decode events:
# config disable_decode_alerts
config enable_decode_drops

# Stop Alerts on experimental TCP options
config disable_tcpopt_experimental_alerts

# Stop Alerts on obsolete TCP options
config disable_tcpopt_obsolete_alerts

# Stop Alerts on T/TCP alerts
config disable_tcpopt_ttcp_alerts

# Stop Alerts on all other TCPOption type events:
config disable_tcpopt_alerts

# Stop Alerts on invalid ip options
```

```
config disable_ipopt_alerts

# Alert if value in length field (IP, TCP, UDP) is greater
th elength of the packet
config enable_decode_oversized_alerts

# Same as above, but drop packet if in Inline mode (requires
enable_decode_oversized_alerts)
config enable_decode_oversized_drops

# Configure IP / TCP checksum mode
config checksum_mode: all

# Configure maximum number of flowbit references.  For more
information, see README.flowbits
# config flowbits_size: 64

# Configure ports to ignore
# config ignore_ports: tcp 21 6667:6671 1356
# config ignore_ports: udp 1:17 53

# Configure active response for non inline operation. For
more information, see REAMDE.active
# config response: eth0 attempts 2

# Configure DAQ related options for inline operation. For
more information, see README.daq
#
config daq: afpacket
config daq_dir: /usr/local/lib/daq
config daq_mode: inline
config policy_mode: inline
# config daq_var: <var>
#
# <type> ::= pcap | afpacket | dump | nfq | ipq | ipfw
# <mode> ::= read-file | passive | inline
# <var> ::= arbitrary <name>=<value passed to DAQ
# <dir> ::= path as to where to look for DAQ module so's

# Configure specific UID and GID to run snort as after
dropping privs. For more information see snort -h command
line options
#
# config set_gid:
# config set_uid:

# Configure default snaplen. Snort defaults to MTU of in use
interface. For more information see README
#
# config snaplen:
#
```

```
# Configure default bpf_file to use for filtering what
traffic reaches snort. For more information see snort -h
command line options (-F)
#
# config bpf_file:
#

# Configure default log directory for snort to log to.  For
more information see snort -h command line options (-l)
#
# config logdir:

##################################################
# Step #3: Configure the base detection engine.  For more
information, see  README.decode
##################################################

# Configure PCRE match limitations
config pcre_match_limit: 3500
config pcre_match_limit_recursion: 1500

# Configure the detection engine  See the Snort Manual,
Configuring Snort - Includes - Config
config detection: search-method ac-split search-optimize
max-pattern-len 20

# Configure the event queue.  For more information, see
README.event_queue
config event_queue: max_queue 8 log 5 order_events
content_length

##################################################
## Configure GTP if it is to be used.
## For more information, see README.GTP
##################################################

# config enable_gtp

##################################################
# Per packet and rule latency enforcement
# For more information see README.ppm
##################################################

# Per Packet latency configuration
#config ppm: max-pkt-time 250, \
#   fastpath-expensive-packets, \
#   pkt-log

# Per Rule latency configuration
#config ppm: max-rule-time 200, \
#   threshold 3, \
#   suspend-expensive-rules, \
#   suspend-timeout 20, \
```

```
#   rule-log alert

##################################################
# Configure Perf Profiling for debugging
# For more information see README.PerfProfiling
##################################################

#config profile_rules: print all, sort avg_ticks
#config profile_preprocs: print all, sort avg_ticks

##################################################
# Configure protocol aware flushing
# For more information see README.stream5
##################################################
config paf_max: 16000

##################################################
# Step #4: Configure dynamic loaded libraries.
# For more information, see Snort Manual, Configuring Snort
- Dynamic Modules
##################################################

# path to dynamic preprocessor libraries
dynamicpreprocessor                                directory
/usr/local/lib/snort_dynamicpreprocessor/

# path to base preprocessor engine
dynamicengine
/usr/local/lib/snort_dynamicengine/libsf_engine.so

# path to dynamic rules libraries
dynamicdetection directory /usr/local/lib/snort_dynamicrules

##################################################
# Step #5: Configure preprocessors
# For more information, see the Snort Manual, Configuring
Snort - Preprocessors
##################################################

# GTP Control Channle Preprocessor. For more information,
see README.GTP
# preprocessor gtp: ports { 2123 3386 2152 }

# Inline packet normalization. For more information, see
README.normalize
# Does nothing in IDS mode
preprocessor normalize_ip4
preprocessor normalize_tcp: ips ecn stream
preprocessor normalize_icmp4
preprocessor normalize_ip6
preprocessor normalize_icmp6
```

```
# Target-based IP defragmentation.  For more inforation, see
README.frag3
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy windows detect_anomalies
overlap_limit 10 min_fragment_length 100 timeout 180

# Target-Based stateful inspection/stream reassembly.  For
more inforation, see README.stream5
preprocessor stream5_global: track_tcp yes, \
   track_udp yes, \
   track_icmp no, \
   max_tcp 262144, \
   max_udp 131072, \
   max_active_responses 2, \
   min_response_seconds 5
preprocessor stream5_tcp: policy windows, detect_anomalies,
require_3whs 180, \
   overlap_limit 10, small_segments 3 bytes 150, timeout
180, \
    ports client 21 22 23 25 42 53 70 79 109 110 111 113 119
135 136 137 139 143 \
        161 445 513 514 587 593 691 1433 1521 1741 2100 3306
6070 6665 6666 6667 6668 6669 \
        7000 8181 32770 32771 32772 32773 32774 32775 32776
32777 32778 32779, \
    ports both 36 80 81 82 83 84 85 86 87 88 89 90 110 311
383 443 465 563 555 591 593 631 636 801 808 818 901 972 989
992 993 994 995 1158 1220 1414 1533 1741 1830 1942 2231 2301
2381 2578 2809 2980 3029 3037 3057 3128 3443 3702 4000 4343
4848 5000 5117 5250 5600 6080 6173 6988 7907 7000 7001 7071
7144 7145 7510 7802 7770 7777 7778 7779 \
        7801 7900 7901 7902 7903 7904 7905 7906 7908 7909
7910 7911 7912 7913 7914 7915 7916 \
        7917 7918 7919 7920 8000 8008 8014 8028 8080 8081
8082 8085 8088 8090 8118 8123 8180 8181 8222 8243 8280 8300
8333 8344 8500 8509 8800 8888 8899 8983 9000 9060 9080 9090
9091 9111 9290 9443 9999 10000 11371 12601 13014 15489 29991
33300 34412 34443 34444 41080 44449 50000 50002 51423 53331
55252 55555 56712
preprocessor stream5_udp: timeout 180

# performance statistics.  For more information, see the
Snort Manual, Configuring Snort - Preprocessors -
Performance Monitor
#    preprocessor    perfmonitor:    time    300    file
/var/snort/snort.stats pktcnt 10000

# HTTP normalization and anomaly detection.  For more
information, see README.http_inspect
preprocessor    http_inspect:    global    iis_unicode_map
unicode.map 1252 compress_depth 65535 decompress_depth 65535
preprocessor http_inspect_server: server default \
```

```
    http_methods { GET POST PUT SEARCH MKCOL COPY MOVE LOCK
UNLOCK NOTIFY POLL BCOPY BDELETE BMOVE LINK UNLINK OPTIONS
HEAD DELETE TRACE TRACK CONNECT SOURCE SUBSCRIBE UNSUBSCRIBE
PROPFIND PROPPATCH BPROPFIND BPROPPATCH RPC_CONNECT
PROXY_SUCCESS BITS_POST CCM_POST SMS_POST RPC_IN_DATA
RPC_OUT_DATA RPC_ECHO_DATA } \
    chunk_length 500000 \
    server_flow_depth 0 \
    client_flow_depth 0 \
    post_depth 65495 \
    oversize_dir_length 500 \
    max_header_length 750 \
    max_headers 100 \
    max_spaces 200 \
    small_chunk_length { 10 5 } \
    ports { 36 80 81 82 83 84 85 86 87 88 89 90 311 383 555
591 593 631 801 808 818 901 972 1158 1220 1414 1533 1741
1830 1942 2231 2301 2381 2578 2809 2980 3029 3037 3057 3128
3443 3702 4000 4343 4848 5000 5117 5250 5600 6080 6173 6988
7000 7001 7071 7144 7145 7510 7770 7777 7778 7779 8000 8008
8014 8028 8080 8081 8082 8085 8088 8090 8118 8123 8180 8181
8222 8243 8280 8300 8333 8344 8500 8509 8800 8888 8899 8983
9000 9060 9080 9090 9091 9111 9290 9443 9999 10000 11371
12601 13014 15489 29991 33300 34412 34443 34444 41080 44449
50000 50002 51423 53331 55252 55555 56712 } \
    non_rfc_char { 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 }
\
    enable_cookie \
    extended_response_inspection \
    inspect_gzip \
    normalize_utf \
    unlimited_decompress \
    normalize_javascript \
    apache_whitespace no \
    ascii no \
    bare_byte no \
    directory no \
    double_decode no \
    iis_backslash no \
    iis_delimiter no \
    iis_unicode no \
    multi_slash no \
    utf_8 no \
    u_encode yes \
    webroot no

# ONC-RPC normalization and anomaly detection.  For more
information, see the Snort Manual, Configuring Snort -
Preprocessors - RPC Decode
preprocessor rpc_decode: 111 32770 32771 32772 32773 32774
32775 32776 32777 32778 32779 no_alert_multiple_requests
no_alert_large_fragments no_alert_incomplete
```

```
# Back Orifice detection.
preprocessor bo

# FTP / Telnet normalization and anomaly detection.  For
more information, see README.ftptelnet
preprocessor ftp_telnet: global inspection_type stateful
encrypted_traffic no check_encrypted
preprocessor ftp_telnet_protocol: telnet \
    ayt_attack_thresh 20 \
    normalize ports { 23 } \
    detect_anomalies
preprocessor ftp_telnet_protocol: ftp server default \
    def_max_param_len 100 \
    ports { 21 2100 3535 } \
    telnet_cmds yes \
    ignore_telnet_erase_cmds yes \
    ftp_cmds { ABOR ACCT ADAT ALLO APPE AUTH CCC CDUP } \
    ftp_cmds { CEL CLNT CMD CONF CWD DELE ENC EPRT } \
    ftp_cmds { EPSV ESTA ESTP FEAT HELP LANG LIST LPRT } \
    ftp_cmds { LPSV MACB MAIL MDTM MIC MKD MLSD MLST } \
    ftp_cmds { MODE NLST NOOP OPTS PASS PASV PBSZ PORT } \
    ftp_cmds { PROT PWD QUIT REIN REST RETR RMD RNFR } \
    ftp_cmds { RNTO SDUP SITE SIZE SMNT STAT STOR STOU } \
    ftp_cmds { STRU SYST TEST TYPE USER XCUP XCRC XCWD } \
    ftp_cmds { XMAS XMD5 XMKD XPWD XRCP XRMD XRSQ XSEM } \
    ftp_cmds { XSEN XSHA1 XSHA256 } \
    alt_max_param_len 0 { ABOR CCC CDUP ESTA FEAT LPSV NOOP
PASV PWD QUIT REIN STOU SYST XCUP XPWD } \
    alt_max_param_len 200 { ALLO APPE CMD HELP NLST RETR
RNFR STOR STOU XMKD } \
    alt_max_param_len 256 { CWD RNTO } \
    alt_max_param_len 400 { PORT } \
    alt_max_param_len 512 { SIZE } \
    chk_str_fmt { ACCT ADAT ALLO APPE AUTH CEL CLNT CMD } \
    chk_str_fmt { CONF CWD DELE ENC EPRT EPSV ESTP HELP } \
    chk_str_fmt { LANG LIST LPRT MACB MAIL MDTM MIC MKD } \
    chk_str_fmt { MLSD MLST MODE NLST OPTS PASS PBSZ PORT }
\
    chk_str_fmt { PROT REST RETR RMD RNFR RNTO SDUP SITE } \
    chk_str_fmt { SIZE SMNT STAT STOR STRU TEST TYPE USER }
\
    chk_str_fmt { XCRC XCWD XMAS XMD5 XMKD XRCP XRMD XRSQ }
\
    chk_str_fmt { XSEM XSEN XSHA1 XSHA256 } \
    cmd_validity ALLO < int [ char R int ] > \
    cmd_validity EPSV < [ { char 12 | char A char L char L }
] > \
    cmd_validity MACB < string > \
    cmd_validity MDTM < [ date nnnnnnnnnnnnnn[.n[n[n]]] ]
string > \
    cmd_validity MODE < char ASBCZ > \
    cmd_validity PORT < host_port > \
    cmd_validity PROT < char CSEP > \
```

```
    cmd_validity STRU < char FRPO [ string ] > \
    cmd_validity TYPE < { char AE [ char NTC ] | char I |
char L [ number ] } >
preprocessor ftp_telnet_protocol: ftp client default \
    max_resp_len 256 \
    bounce yes \
    ignore_telnet_erase_cmds yes \
    telnet_cmds yes

# SMTP normalization and anomaly detection.   For more
information, see README.SMTP
preprocessor smtp: ports { 25 465 587 691 } \
    inspection_type stateful \
    b64_decode_depth 0 \
    qp_decode_depth 0 \
    bitenc_decode_depth 0 \
    uu_decode_depth 0 \
    log_mailfrom \
    log_rcptto \
    log_filename \
    log_email_hdrs \
    normalize_cmds \
    normalize_cmds { ATRN AUTH BDAT CHUNKING DATA DEBUG EHLO
EMAL ESAM ESND ESOM ETRN EVFY } \
    normalize_cmds { EXPN HELO HELP IDENT MAIL NOOP ONEX
QUEU QUIT RCPT RSET SAML SEND SOML } \
    normalize_cmds { STARTTLS TICK TIME TURN TURNME VERB
VRFY X-ADAT X-DRCP X-ERCP X-EXCH50 } \
    normalize_cmds { X-EXPS X-LINK2STATE XADR XAUTH XCIR
XEXCH50 XGEN XLICENSE XQUE XSTA XTRN XUSR } \
    max_command_line_len 512 \
    max_header_line_len 1000 \
    max_response_line_len 512 \
    alt_max_command_line_len 260 { MAIL } \
    alt_max_command_line_len 300 { RCPT } \
    alt_max_command_line_len 500 { HELP HELO ETRN EHLO } \
    alt_max_command_line_len 255 { EXPN VRFY ATRN SIZE BDAT
DEBUG EMAL ESAM ESND ESOM EVFY IDENT NOOP RSET } \
    alt_max_command_line_len 246 { SEND SAML SOML AUTH TURN
ETRN DATA RSET QUIT ONEX QUEU STARTTLS TICK TIME TURNME VERB
X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50 XGEN XLICENSE
XQUE XSTA XTRN XUSR } \
    valid_cmds { ATRN AUTH BDAT CHUNKING DATA DEBUG EHLO
EMAL ESAM ESND ESOM ETRN EVFY } \
    valid_cmds { EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU
QUIT RCPT RSET SAML SEND SOML } \
    valid_cmds { STARTTLS TICK TIME TURN TURNME VERB VRFY X-
ADAT X-DRCP X-ERCP X-EXCH50 } \
    valid_cmds { X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50
XGEN XLICENSE XQUE XSTA XTRN XUSR } \
    xlink2state { enabled }
```

```
# Portscan detection. For more information, see
README.sfportscan
preprocessor sfportscan: proto { all } memcap { 10000000 }
sense_level { low } scan_type { all }

# ARP spoof detection. For more information, see the Snort
Manual - Configuring Snort - Preprocessors - ARP Spoof
Preprocessor
# preprocessor arpspoof
# preprocessor arpspoof_detect_host: 192.168.40.1
f0:0f:00:f0:0f:00

# SSH anomaly detection. For more information, see
README.ssh
preprocessor ssh: server_ports { 22 } \
                  autodetect \
                  max_client_bytes 19600 \
                  max_encrypted_packets 20 \
                  max_server_version_len 100 \
                  enable_respoverflow enable_ssh1crc32 \
                  enable_srvoverflow enable_protomismatch

# SMB / DCE-RPC normalization and anomaly detection. For
more information, see README.dcerpc2
preprocessor dcerpc2: memcap 102400, events [co ]
preprocessor dcerpc2_server: default, policy WinXP, \
    detect [smb [139,445], tcp 135, udp 135, rpc-over-http-
server 593], \
    autodetect [tcp 1025:, udp 1025:, rpc-over-http-server
1025:], \
    smb_max_chain 3, smb_invalid_shares ["C$", "D$",
"ADMIN$"]

# DNS anomaly detection. For more information, see
README.dns
preprocessor dns: ports { 53 } enable_rdata_overflow

# SSL anomaly detection and traffic bypass. For more
information, see README.ssl
preprocessor ssl: ports { 443 465 563 636 989 992 993 994
995 5061 7801 7802 7900 7901 7902 7903 7904 7905 7906 7907
7908 7909 7910 7911 7912 7913 7914 7915 7916 7917 7918 7919
7920 }, trustservers, noinspect_encrypted

# SDF sensitive data preprocessor. For more information see
README.sensitive_data
preprocessor sensitive_data: alert_threshold 25

# SIP Session Initiation Protocol preprocessor. For more
information see README.sip
preprocessor sip: max_sessions 40000, \
   ports { 5060 5061 5600 }, \
   methods { invite \
```

```
                cancel \
                ack \
                bye \
                register \
                options \
                refer \
                subscribe \
                update \
                join \
                info \
                message \
                notify \
                benotify \
                do \
                qauth \
                sprack \
                publish \
                service \
                unsubscribe \
                prack }, \
   max_uri_len 512, \
   max_call_id_len 80, \
   max_requestName_len 20, \
   max_from_len 256, \
   max_to_len 256, \
   max_via_len 1024, \
   max_contact_len 512, \
   max_content_len 2048

# IMAP preprocessor.  For more information see README.imap
preprocessor imap: \
   ports { 143 } \
   b64_decode_depth 0 \
   qp_decode_depth 0 \
   bitenc_decode_depth 0 \
   uu_decode_depth 0

# POP preprocessor. For more information see README.pop
preprocessor pop: \
   ports { 110 } \
   b64_decode_depth 0 \
   qp_decode_depth 0 \
   bitenc_decode_depth 0 \
   uu_decode_depth 0

#   Modbus   preprocessor.   For   more   information   see
README.modbus
preprocessor modbus: ports { 502 }

# DNP3 preprocessor. For more information see README.dnp3
preprocessor dnp3: ports { 20000 } \
   memcap 262144 \
   check_crc
```

```
# Reputation preprocessor. For more information see
README.reputation
#preprocessor reputation: \
#   memcap 500, \
#   priority whitelist, \
#   nested_ip inner, \
#   whitelist $WHITE_LIST_PATH/white_list.rules, \
#   blacklist $BLACK_LIST_PATH/black_list.rules

##################################################
# Step #6: Configure output plugins
# For more information, see Snort Manual, Configuring Snort
- Output Modules
##################################################

# unified2
# Recommended for most installs
# output unified2: filename merged.log, limit 128, nostamp,
mpls_event_types, vlan_event_types
output unified2: filename merged.log, limit 128

# Additional configuration for specific types of installs
# output alert_unified2: filename snort.alert, limit 128,
nostamp
# output log_unified2: filename snort.log, limit 128,
nostamp

# syslog
# output alert_syslog: LOG_AUTH LOG_ALERT

# pcap
# output log_tcpdump: tcpdump.log

# metadata reference data.  do not modify these lines
include classification.config
include reference.config

##################################################
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort
Rules
#
# NOTE: All categories are enabled in this conf file
##################################################

# site specific rules
include $RULE_PATH/local.rules
include $RULE_PATH/snort.rules
include $RULE_PATH/so_rules.rules

# include $RULE_PATH/app-detect.rules
# include $RULE_PATH/attack-responses.rules
```

```
# include $RULE_PATH/backdoor.rules
# include $RULE_PATH/bad-traffic.rules
# include $RULE_PATH/blacklist.rules
# include $RULE_PATH/botnet-cnc.rules
# include $RULE_PATH/browser-chrome.rules
# include $RULE_PATH/browser-firefox.rules
# include $RULE_PATH/browser-ie.rules
# include $RULE_PATH/browser-other.rules
# include $RULE_PATH/browser-plugins.rules
# include $RULE_PATH/browser-webkit.rules
# include $RULE_PATH/chat.rules
# include $RULE_PATH/content-replace.rules
# include $RULE_PATH/ddos.rules
# include $RULE_PATH/dns.rules
# include $RULE_PATH/dos.rules
# include $RULE_PATH/experimental.rules
# include $RULE_PATH/exploit-kit.rules
# include $RULE_PATH/exploit.rules
# include $RULE_PATH/file-executable.rules
# include $RULE_PATH/file-flash.rules
# include $RULE_PATH/file-identify.rules
# include $RULE_PATH/file-image.rules
# include $RULE_PATH/file-java.rules
# include $RULE_PATH/file-multimedia.rules
# include $RULE_PATH/file-office.rules
# include $RULE_PATH/file-other.rules
# include $RULE_PATH/file-pdf.rules
# include $RULE_PATH/finger.rules
# include $RULE_PATH/ftp.rules
# include $RULE_PATH/icmp-info.rules
# include $RULE_PATH/icmp.rules
# include $RULE_PATH/imap.rules
# include $RULE_PATH/indicator-compromise.rules
# include $RULE_PATH/indicator-obfuscation.rules
# include $RULE_PATH/indicator-scan.rules
# include $RULE_PATH/indicator-shellcode.rules
# include $RULE_PATH/info.rules
# include $RULE_PATH/malware-backdoor.rules
# include $RULE_PATH/malware-cnc.rules
# include $RULE_PATH/malware-other.rules
# include $RULE_PATH/malware-tools.rules
# include $RULE_PATH/misc.rules
# include $RULE_PATH/multimedia.rules
# include $RULE_PATH/mysql.rules
# include $RULE_PATH/netbios.rules
# include $RULE_PATH/nntp.rules
# include $RULE_PATH/oracle.rules
# include $RULE_PATH/os-linux.rules
# include $RULE_PATH/os-mobile.rules
# include $RULE_PATH/os-other.rules
# include $RULE_PATH/os-solaris.rules
# include $RULE_PATH/os-windows.rules
# include $RULE_PATH/other-ids.rules
```

```
# include $RULE_PATH/p2p.rules
# include $RULE_PATH/phishing-spam.rules
# include $RULE_PATH/policy-multimedia.rules
# include $RULE_PATH/policy-other.rules
# include $RULE_PATH/policy.rules
# include $RULE_PATH/policy-social.rules
# include $RULE_PATH/policy-spam.rules
# include $RULE_PATH/pop2.rules
# include $RULE_PATH/pop3.rules
# include $RULE_PATH/protocol-dns.rules
# include $RULE_PATH/protocol-finger.rules
# include $RULE_PATH/protocol-ftp.rules
# include $RULE_PATH/protocol-icmp.rules
# include $RULE_PATH/protocol-imap.rules
# include $RULE_PATH/protocol-nntp.rules
# include $RULE_PATH/protocol-other.rules
# include $RULE_PATH/protocol-pop.rules
# include $RULE_PATH/protocol-rpc.rules
# include $RULE_PATH/protocol-scada.rules
# include $RULE_PATH/protocol-services.rules
# include $RULE_PATH/protocol-snmp.rules
# include $RULE_PATH/protocol-telnet.rules
# include $RULE_PATH/protocol-tftp.rules
# include $RULE_PATH/protocol-voip.rules
# include $RULE_PATH/pua-adware.rules
# include $RULE_PATH/pua-other.rules
# include $RULE_PATH/pua-p2p.rules
# include $RULE_PATH/pua-toolbars.rules
# include $RULE_PATH/rpc.rules
# include $RULE_PATH/rservices.rules
# include $RULE_PATH/scada.rules
# include $RULE_PATH/scan.rules
# include $RULE_PATH/server-apache.rules
# include $RULE_PATH/server-iis.rules
# include $RULE_PATH/server-mail.rules
# include $RULE_PATH/server-mssql.rules
# include $RULE_PATH/server-mysql.rules
# include $RULE_PATH/server-oracle.rules
# include $RULE_PATH/server-other.rules
# include $RULE_PATH/server-samba.rules
# include $RULE_PATH/server-webapp.rules
# include $RULE_PATH/shellcode.rules
# include $RULE_PATH/smtp.rules
# include $RULE_PATH/snmp.rules
# include $RULE_PATH/specific-threats.rules
# include $RULE_PATH/spyware-put.rules
# include $RULE_PATH/sql.rules
# include $RULE_PATH/telnet.rules
# include $RULE_PATH/tftp.rules
# include $RULE_PATH/virus.rules
# include $RULE_PATH/voip.rules
# include $RULE_PATH/web-activex.rules
# include $RULE_PATH/web-attacks.rules
```

```
# include $RULE_PATH/web-cgi.rules
# include $RULE_PATH/web-client.rules
# include $RULE_PATH/web-coldfusion.rules
# include $RULE_PATH/web-frontpage.rules
# include $RULE_PATH/web-iis.rules
# include $RULE_PATH/web-misc.rules
# include $RULE_PATH/web-php.rules
# include $RULE_PATH/x11.rules

##################################################
# Step #8: Customize your preprocessor and decoder alerts
# For more information, see README.decoder_preproc_rules
##################################################

# decoder and preprocessor event rules
#include $PREPROC_RULE_PATH/preprocessor.rules
#include $PREPROC_RULE_PATH/decoder.rules
#include $PREPROC_RULE_PATH/sensitive-data.rules

##################################################
# Step #9: Customize your Shared Object Snort Rules
#     For     more     information,     see     http://vrt-
blog.snort.org/2009/01/using-vrt-certified-shared-object-
rules.html
##################################################

# dynamic library rules
# include $SO_RULE_PATH/browser-ie.rules
# include $SO_RULE_PATH/browser-other.rules
# include $SO_RULE_PATH/browser-plugins.rules
# include $SO_RULE_PATH/exploit-kit.rules
# include $SO_RULE_PATH/file-executable.rules
# include $SO_RULE_PATH/file-flash.rules
# include $SO_RULE_PATH/file-image.rules
# include $SO_RULE_PATH/file-java.rules
# include $SO_RULE_PATH/file-multimedia.rules
# include $SO_RULE_PATH/file-office.rules
# include $SO_RULE_PATH/file-other.rules
# include $SO_RULE_PATH/file-pdf.rules
# include $SO_RULE_PATH/indicator-shellcode.rules
# include $SO_RULE_PATH/malware-cnc.rules
# include $SO_RULE_PATH/malware-other.rules
# include $SO_RULE_PATH/netbios.rules
# include $SO_RULE_PATH/os-linux.rules
# include $SO_RULE_PATH/os-other.rules
# include $SO_RULE_PATH/os-windows.rules
# include $SO_RULE_PATH/policy-social.rules
# include $SO_RULE_PATH/protocol-dns.rules
# include $SO_RULE_PATH/protocol-icmp.rules
# include $SO_RULE_PATH/protocol-nntp.rules
# include $SO_RULE_PATH/protocol-other.rules
# include $SO_RULE_PATH/protocol-snmp.rules
# include $SO_RULE_PATH/protocol-voip.rules
```

```
# include $SO_RULE_PATH/pua-p2p.rules
# include $SO_RULE_PATH/server-apache.rules
# include $SO_RULE_PATH/server-iis.rules
# include $SO_RULE_PATH/server-mail.rules
# include $SO_RULE_PATH/server-mysql.rules
# include $SO_RULE_PATH/server-oracle.rules
# include $SO_RULE_PATH/server-other.rules
# include $SO_RULE_PATH/server-webapp.rules


# legacy dynamic library rule files
# include $SO_RULE_PATH/bad-traffic.rules
# include $SO_RULE_PATH/browser-ie.rules
# include $SO_RULE_PATH/chat.rules
# include $SO_RULE_PATH/dos.rules
# include $SO_RULE_PATH/exploit.rules
# include $SO_RULE_PATH/file-flash.rules
# include $SO_RULE_PATH/icmp.rules
# include $SO_RULE_PATH/imap.rules
# include $SO_RULE_PATH/misc.rules
# include $SO_RULE_PATH/multimedia.rules
# include $SO_RULE_PATH/netbios.rules
# include $SO_RULE_PATH/nntp.rules
# include $SO_RULE_PATH/p2p.rules
# include $SO_RULE_PATH/smtp.rules
# include $SO_RULE_PATH/snmp.rules
# include $SO_RULE_PATH/specific-threats.rules
# include $SO_RULE_PATH/web-activex.rules
# include $SO_RULE_PATH/web-client.rules
# include $SO_RULE_PATH/web-iis.rules
# include $SO_RULE_PATH/web-misc.rules

#   Event   thresholding   or   suppression   commands.   See
threshold.conf
include threshold.conf
```

Lampiran 3. ***Snort Startup Scripts*** **(snortd)**

```
#!/bin/sh
# $Id$
#
# snortd         Start/Stop the snort IDS daemon.
#
# chkconfig: 2345 99 99
# description:   snort is a lightweight network intrusion
detection tool that \
#                 currently detects more than 1100 host and
network \
#                vulnerabilities, portscans, backdoors, and
more.
#

# Source function library.
. /etc/rc.d/init.d/functions

# Source the local configuration file
. /etc/sysconfig/snort

# Variable for loading and unloading bridge
IFCONFIG=/sbin/ifconfig

# Convert the /etc/sysconfig/snort settings to something
snort can
# use on the startup line.
if [ "$ALERTMODE"X = "X" ]; then
   ALERTMODE=""
else
   ALERTMODE="-A $ALERTMODE"
fi

if [ "$USER"X = "X" ]; then
   USER="snort"
fi

if [ "$GROUP"X = "X" ]; then
   GROUP="snort"
fi

if [ "$BINARY_LOG"X = "1X" ]; then
   BINARY_LOG="-b"
else
   BINARY_LOG=""
fi

if [ "$CONF"X = "X" ]; then
   CONF="-c /etc/snort/snort.conf"
else
   CONF="-c $CONF"
fi
```

```
if [ "$INTERFACE"X = "X" ]; then
   INTERFACE="-i eth0"
else
   INTERFACE="-i $INTERFACE"
fi

if [ "$DUMP_APP"X = "1X" ]; then
   DUMP_APP="-d"
else
   DUMP_APP=""
fi

if [ "$NO_PACKET_LOG"X = "1X" ]; then
   NO_PACKET_LOG="-N"
else
   NO_PACKET_LOG=""
fi

if [ "$PRINT_INTERFACE"X = "1X" ]; then
   PRINT_INTERFACE="-I"
else
   PRINT_INTERFACE=""
fi

if [ "$PASS_FIRST"X = "1X" ]; then
   PASS_FIRST="-o"
else
   PASS_FIRST=""
fi

if [ "$LOGDIR"X = "X" ]; then
   LOGDIR=/var/log/snort
fi

# These are used by the 'stats' option
if [ "$SYSLOG"X = "X" ]; then
   SYSLOG=/var/log/messages
fi

if [ "$SECS"X = "X" ]; then
   SECS=5
fi

if [ ! "$BPFFILE"X = "X" ]; then
   BPFFILE="-F $BPFFILE"
fi

# Pass packets to target QUEUE if using snort in inline mode
if [ "$QUEUE"X = "1X" ]; then
   QUEUE="-Q"
else
   QUEUE=""
```

```
fi

#####################################
# Now to the real heart of the matter:

# See how we were called.
case "$1" in
  start)
      ## Bring the bridge down if is up
      ## so it will not conflict with the DAQ
      $IFCONFIG br0 down
         echo -n "Starting snort: "
         cd $LOGDIR
         if [ "$INTERFACE" = "-i ALL" ]; then
            for i in `cat /proc/net/dev|grep eth|awk -F ":"
'{ print $1; }'`
            do
                 mkdir -p "$LOGDIR/$i"
                 chown -R $USER:$GROUP $LOGDIR
                 daemon  /usr/sbin/snort  $QUEUE  $ALERTMODE
$BINARY_LOG $NO_PACKET_LOG $DUMP_APP -D $PRINT_INTERFACE -i
$i  -u  $USER  -g  $GROUP  $CONF  -l  $LOGDIR/$i  $PASS_FIRST
$BPFFILE $BPF
            done
         else
            # check if more than one interface is given
            if [ `echo $INTERFACE|wc -w` -gt 2 ]; then
               for i in `echo $INTERFACE | sed s/"-i "//`
                 do
                   mkdir -p "$LOGDIR/$i"
                   chown -R $USER:$GROUP $LOGDIR
                   daemon  /usr/sbin/snort  $QUEUE  $ALERTMODE
$BINARY_LOG $NO_PACKET_LOG $DUMP_APP -D $PRINT_INTERFACE -i
$i  -u  $USER  -g  $GROUP  $CONF  -l  $LOGDIR/$i  $PASS_FIRST
$BPFFILE $BPF
              done
            else
               # Run with a single interface (default)
               daemon   /usr/sbin/snort   $QUEUE   $ALERTMODE
$BINARY_LOG  $NO_PACKET_LOG  $DUMP_APP  -D  $PRINT_INTERFACE
$INTERFACE -u $USER -g $GROUP $CONF -l $LOGDIR $PASS_FIRST
$BPFFILE $BPF
            fi
         fi
         touch /var/lock/subsys/snort
         echo
         ;;
  stop)
         echo -n "Stopping snort: "
         killproc snort
         rm -f /var/lock/subsys/snort
         echo
      $IFCONFIG br0 up -arp
```

```
        ;;
  reload)
        echo "Sorry, not implemented yet"
        ;;
  restart)
        $0 stop
        $0 start
        ;;
  condrestart)
        [ -e /var/lock/subsys/snort ] && $0 restart
        ;;
  status)
        status snort
        ;;
  stats)
        TC=125                          # Trailing context
to grep
        SNORTNAME='snort'                # Process name to
look for

        if [ ! -x "/sbin/pidof" ]; then
           echo "/sbin/pidof not present, sorry, I cannot go
on like this!"
           exit 1
        fi

        #Grab Snort's PID
        PID=`pidof -o $$ -o $PPID -o %PPID -x ${SNORTNAME}`

        if [ ! -n "$PID" ]; then         # if we got no PID
then:
           echo  "No  PID  found:  ${SNORTNAME}  must  not
running."
           exit 2
        fi

        echo ""
        echo "*******"
        echo  "WARNING:   This  feature  is  EXPERIMENTAL  -
please report errors!"
        echo "*******"
        echo ""
        echo "You can also run: $0 stats [long | opt]"
        echo ""
        echo "Dumping ${SNORTNAME}'s ($PID) statistics"
        echo "please wait..."

        # Get the date and tell Snort to dump stats as close
together in
        # time as possible--not 100%, but it seems to work.
        startdate=`date '+%b %e %H:%M:%S'`

        # This causes the stats to be dumped to syslog
```

```
        kill -USR1 $PID

        # Sleep for $SECS secs to give syslog a chance to
catch up
        # May need to be adjusted for slow/busy systems
        sleep $SECS

        if [ "$2" = "long" ]; then                  # Long
format
            egrep -B 3 -A $TC "^$startdate .* snort.*:
={79}" $SYSLOG | \
                grep snort.*:
        elif [ "$2" = "opt" ]; then              # OPTimize
format
           # Just show stuff useful for optimizing Snort
            egrep -B 3 -A $TC "^$startdate .* snort.*:
={79}" $SYSLOG | \
                egrep "snort.*: Snort analyzed |snort.*:
dropping|emory .aults:"
        else                                        # Default
format
            egrep -B 3 -A $TC "^$startdate .* snort.*:
={79}" $SYSLOG | \
                grep snort.*: | cut -d: -f4-
        fi
        ;;
  *)
        echo                    "Usage:                  $0
{start|stop|reload|restart|condrestart|status|stats
(long|opt)}"
        exit 2
esac

exit 0
```

Lampiran 4. File Konfigurasi Barnyard2 (barnyard.conf)

```
#  Barnyard2 example configuration file

# This file contains a sample barnyard2 configuration.
# You can take the following steps to create your own custom
configuration:

#   1) Configure the variable declarations
#   2) Setup the input plugins
#   3) Setup the output plugins

# Step 1: configure the variable declarations

# in order to keep from having a commandline that uses every
letter in the
# alphabet most configuration options are set here.

# use UTC for timestamps
#config utc

# set the appropriate paths to the file(s) your Snort
process is using.

config reference_file:      /etc/snort/reference.config
config classification_file: /etc/snort/classification.config
config gen_file:            /etc/snort/gen-msg.map
config sid_file:            /etc/snort/sid-msg.map

# Configure signature suppression at the spooler level see
doc/README.sig_suppress
#config sig_suppress: 1:10

# Set the event cache size to defined max value before
recycling of event occur.
#config event_cache_size: 4096

# define dedicated references similar to that of snort.

#config reference: mybugs http://www.mybugs.com/?s=

# define explicit classifications similar to that of snort.

#config classification: shortname, short description,
priority

# set the directory for any output logging
#config logdir: /tmp

# to ensure that any plugins requiring some level of
uniqueness in their output
# the alert_with_interface_name, interface and hostname
directives are provided.
```

```
# An example of usage would be to configure them to the
values of the associated
# snort process whose unified files you are reading.
#
# Example:
#   For a snort process as follows:
#     snort -i eth0 -c /etc/snort.conf
#
#   Typical options would be:
#     config hostname:  thor
#     config interface: eth0
#     config alert_with_interface_name
#
config hostname:   IPSSVR01
config interface:  eth1:eth2

# enable printing of the interface name when alerting.
#
#config alert_with_interface_name

# at times snort will alert on a packet within a stream and
dump that stream to
# the unified output. barnyard2 can generate output on each
packet of that
# stream or the first packet only.
#
#config alert_on_each_packet_in_stream

# enable daemon mode
#config daemon

# make barnyard2 process chroot to directory after
initialisation.
#config chroot: /var/spool/barnyard2

# specifiy the group or GID for barnyard2 to run as after
initialisation.
#config set_gid: 999

# specifiy the user or UID for barnyard2 to run as after
initialisation.
#config set_uid: 999

# specify the directory for the barnyard2 PID file.
#config pidpath: /var/run/by2.pid

# enable decoding of the data link (or second level
headers).
#config decode_data_link

# dump the application data
#config dump_payload
```

```
# dump the application data as chars only
#config dump_chars_only

# enable verbose dumping of payload information in log style
output plugins.
#config dump_payload_verbose

# enable obfuscation of logged IP addresses.
#config obfuscate

# enable the year being shown in timestamps
#config show_year

# set the umask for all files created by the barnyard2
process (eg. log files).
#config umask: 066

# enable verbose logging
#config verbose

# quiet down some of the output
#config quiet

# define the full waldo filepath.
#config waldo_file: /tmp/waldo
config waldo_file: /var/log/snort/eth1/barnyard2.waldo


# specificy the maximum length of the MPLS label chain
#config max_mpls_labelchain_len: 64

# specify the protocol (ie ipv4, ipv6, ethernet) that is
encapsulated by MPLS.
#config mpls_payload_type: ipv4

# set the reference network or homenet which is
predominantly used by the
# log_ascii plugin.
#config reference_net: 192.168.0.0/24

# CONTINOUS MODE

# set the archive directory for use with continous mode
#config archivedir: /tmp
config archivedir: /var/log/snort/eth1/archive

# when in operating in continous mode, only process new
records and ignore any
# existing unified files
config process_new_records_only

# Step 2: setup the input plugins
```

```
# this is not hard, only unified2 is supported ;)
input unified2

# Step 3: setup the output plugins

# alert_cef
# ------------------------------------------------------------
------------------
#
# Purpose:
#   This output module provides the abilty to output alert
information to a
# remote network host as well as the local host using the
open standard
# Common Event Format (CEF).
#
# Arguments: host=hostname[:port], severity facility
#            arguments should be comma delimited.
#    host           - specify a remote hostname or IP with
optional port number
#                  this is only specific to WIN32 (and is not
yet fully supported)
#    severity      - as defined in RFC 3164 (eg. LOG_WARN,
LOG_INFO)
#    facility      - as defined in RFC 3164 (eg. LOG_AUTH,
LOG_LOCAL0)
#
# Examples:
#   output alert_cef
#   output alert_cef: host=192.168.10.1
#   output alert_cef: host=sysserver.com:1001
#   output alert_cef: LOG_AUTH LOG_INFO
#

# alert_bro
# ------------------------------------------------------------
------------------
#
# Purpose: Send alerts to a Bro-IDS instance.
#
# Arguments: hostname:port
#
# Examples:
#   output alert_bro: 127.0.0.1:47757

# alert_fast
# ------------------------------------------------------------
------------------
#  Purpose: Converts data to an approximation of Snort's
"fast alert" mode.
#
# Arguments: file <file>, stdout
#            arguments should be comma delimited.
```

```
#   file - specifiy alert file
#   stdout - no alert file, just print to screen
#
# Examples:
#   output alert_fast
#   output alert_fast: stdout
#
# output alert_fast: stdout

# prelude: log to the Prelude Hybrid IDS system
# ------------------------------------------------------------
------------------
#
# Purpose:
#  This output module provides logging to the Prelude Hybrid
IDS system
#
# Arguments: profile=snort-profile
#    snort-profile    - name of the Prelude profile to use
(default is snort).
#
# Snort priority to IDMEF severity mappings:
# high < medium < low < info
#
# These are the default mapped from classification.config:
# info   = 4
# low    = 3
# medium = 2
# high   = anything below medium

# Examples:
#   output alert_prelude
#   output alert_prelude: profile=snort-profile-name
#

# alert_syslog
# ------------------------------------------------------------
------------------
#
# Purpose:
#  This output module provides the abilty to output alert
information to local syslog
#
#    severity     - as defined in RFC 3164 (eg. LOG_WARN,
LOG_INFO)
#    facility     - as defined in RFC 3164 (eg. LOG_AUTH,
LOG_LOCAL0)
#
# Examples:
#   output alert_syslog
#   output alert_syslog: LOG_AUTH LOG_INFO
#
```

```
# syslog_full
#--------------------------------
# Available as both a log and alert output plugin.  Used to
output data via TCP/UDP or LOCAL ie(syslog())
# Arguments:
#      sensor_name $sensor_name         - unique sensor name
#      server $server                    - server the device
will report to
#      local                             - if defined, ignore
all remote information and use syslog() to send message.
#       protocol $protocol                 - protocol device
will report over (tcp/udp)
#      port $port                          - destination port
device will report to (default: 514)
#      delimiters $delimiters            - define a character
that will delimit message sections ex:  "|", will use | as
message section delimiters. (default: |)
#        separators $separators               - define field
separator included in each message ex: " " ,  will use space
as field separator.              (default: [:space:])
#      operation_mode $operaion_mode    - default | complete
: default mode is compatible with default snort syslog
message, complete prints more information such as the raw
packet (hexed)
#        log_priority    $log_priority      - used by local
option for syslog priority call. (man syslog(3) for
supported options) (default: LOG_INFO)
#        log_facility  $log_facility        - used by local
option for syslog facility call. (man syslog(3) for
supported options) (default: LOG_USER)
#       payload_encoding                    - (default: hex)
support   hex/ascii/base64   for   log_syslog_full   using
operation_mode complete only.

# Usage Examples:
#  output  alert_syslog_full:  sensor_name  snortIds1-eth2,
server   xxx.xxx.xxx.xxx,   protocol   udp,   port   514,
operation_mode default
#  output  alert_syslog_full:  sensor_name  snortIds1-eth2,
server   xxx.xxx.xxx.xxx,   protocol   udp,   port   514,
operation_mode complete
# output log_syslog_full: sensor_name snortIds1-eth2, server
xxx.xxx.xxx.xxx,  protocol  udp,  port  514,  operation_mode
default
# output log_syslog_full: sensor_name snortIds1-eth2, server
xxx.xxx.xxx.xxx,  protocol  udp,  port  514,  operation_mode
complete
#  output  alert_syslog_full:  sensor_name  snortIds1-eth2,
server xxx.xxx.xxx.xxx, protocol udp, port 514
# output log_syslog_full: sensor_name snortIds1-eth2, server
xxx.xxx.xxx.xxx, protocol udp, port 514
#  output  alert_syslog_full:  sensor_name  snortIds1-eth2,
local
```

```
# output log_syslog_full: sensor_name snortIds1-eth2, local,
log_priority LOG_CRIT,log_facility LOG_CRON

# log_ascii
# ------------------------------------------------------------
------------------
#
# Purpose: This output module provides the default packet
logging funtionality
#
# Arguments: None.
#
# Examples:
#   output log_ascii
#

# log_tcpdump
# ------------------------------------------------------------
------------------
#
# Purpose
#  This output module logs packets in binary tcpdump format
#
# Arguments:
#   The only argument is the output file name.
#
# Examples:
#   output log_tcpdump: tcpdump.log
#

# sguil
# ------------------------------------------------------------
------------------
#
# Purpose: This output module provides logging ability for
the sguil interface
# See doc/README.sguil
#
# Arguments: agent_port <port>, sensor_name <name>
#            arguments should be comma delimited.
#   agent_port  - explicitly set the sguil agent listening
port
#                 (default: 7736)
#   sensor_name - explicitly set the sensor name
#                 (default: machine hostname)
#
# Examples:
#   output sguil
#   output sguil: agent_port=7000
#   output sguil: sensor_name=argyle
#   output sguil: agent_port=7000, sensor_name=argyle
#
```

```
# database: log to a variety of databases
# ------------------------------------------------------------
------------------
#
# Purpose: This output module provides logging ability to a
variety of databases
# See doc/README.database for additional information.
#
# Examples:
#    output database: log, mysql, user=root password=test
dbname=db host=localhost
#       output  database:  alert,  postgresql,  user=snort
dbname=snort
#   output database: log, odbc, user=snort dbname=snort
#    output database: log, mssql, dbname=snort user=snort
password=test
#    output database: log, oracle, dbname=snort user=snort
password=test
#
output database: log, mysql, user=snort password=5235107372
dbname=snort host=localhost

#  alert_fwsam:  allow  blocking  of  IP's  through  remote
services
# ------------------------------------------------------------
------------------
# output alert_fwsam: <SnortSam Station>:<port>/<key>
#
#  <FW Mgmt Station>:  IP address or host name of the host
running SnortSam.
#  <port>:          Port the remote SnortSam service listens
on (default 898).
#   <key>:                      Key  used  for  authentication
(encryption really)
#              of the communication to the remote service.
#
# Examples:
#
# output alert_fwsam: snortsambox/idspassword
# output alert_fwsam: fw1.domain.tld:898/mykey
#       output       alert_fwsam:       192.168.0.1/borderfw
192.168.1.254/wanfw
```

Lampiran 5. File Konfigurasi Pulledpork (pulledpork.conf)

```
# Config file for pulledpork
# Be sure to read through the entire configuration file
# If you specify any of these items on the command line, it
WILL take
# precedence over any value that you specify in this file!

#######
#######   The below section defines what your oinkcode is
(required for
#######  VRT rules), defines a temp path (must be writable)
and also
#######  defines what version of rules that you are getting
(for your
#######  snort version and subscription etc...)
#######

# You can specify one or as many rule_urls as you like, they
#               must               appear              as
http://what.site.com/|rulesfile.tar.gz|1234567.    You  can
specify
# each on an individual line, or you can specify them in a ,
separated list
#                                                       i.e.
rule_url=http://x.y.z/|a.tar.gz|123,http://z.y.z/|b.tar.gz|4
56
# note that the url, rule file, and oinkcode itself are
separated by a pipe |
# i.e. url|tarball|123456789,
rule_url=https://www.snort.org/reg-rules/|snortrules-
snapshot.tar.gz|9df97c6eb0744c5156b4aff4ae9cc22136648eb4
# NEW Community ruleset:
rule_url=https://s3.amazonaws.com/snort-
org/www/rules/community/|community-rules.tar.gz|Community
#   NEW   For   IP   Blacklisting!   Note   the   format   is
urltofile|IPBLACKLIST|<oinkcode>
# This format MUST be followed to let pulledpork know that
this is a blacklist
#rule_url=http://labs.snort.org/feeds/ip-
filter.blf|IPBLACKLIST|open
# URL for rule documentation! (slow to process)
rule_url=https://www.snort.org/reg-
rules/|opensource.gz|9df97c6eb0744c5156b4aff4ae9cc22136648eb
4
#rule_url=https://rules.emergingthreatspro.com/|emerging.rul
es.tar.gz|open
# THE FOLLOWING URL is for etpro downloads, note the tarball
name change!
# and the et oinkcode requirement!
```

```
#rule_url=https://rules.emergingthreatspro.com/|etpro.rules.
tar.gz|<et oinkcode>
# NOTE above that the VRT snortrules-snapshot does not
contain the version
# portion of the tarball name, this is because PP now
automatically populates
# this value for you, if, however you put the version
information in, PP will
# NOT populate this value but will use your value!

# Specify rule categories to ignore from the tarball in a
comma separated list
# with no spaces.  There are four ways to do this:
# 1) Specify the category name with no suffix at all to
ignore the category
#    regardless of what rule-type it is, ie: netbios
# 2) Specify the category name with a '.rules' suffix to
ignore only gid 1
#      rulefiles located in the /rules directory of the
tarball, ie: policy.rules
# 3) Specify the category name with a '.preproc' suffix to
ignore only
#       preprocessor rules located in the /preproc_rules
directory of the tarball,
#    ie: sensitive-data.preproc
# 4) Specify the category name with a '.so' suffix to ignore
only shared-object
#       rules located in the /so_rules directory of the
tarball, ie: netbios.so
# The example below ignores dos rules wherever they may
appear, sensitive-
# data preprocessor rules, p2p so-rules (while including gid
1 p2p rules),
# and netbios gid-1 rules (while including netbios so-
rules):
# ignore = dos,sensitive-data.preproc,p2p.so,netbios.rules
# These defaults are reasonable for the VRT ruleset with
Snort 2.9.0.x.
ignore=deleted.rules,experimental.rules,local.rules
# IMPORTANT, if you are NOT yet using 2.8.6 then you MUST
comment out the
# previous ignore line and uncomment the following!
#
ignore=deleted,experimental,local,decoder,preprocessor,sensi
tive-data

# What is our temp path, be sure this path has a bit of
space for rule
# extraction and manipulation, no trailing slash
temp_path=/tmp

#######
```

```
#######  The below section is for rule processing.  This
section is
#######     required  if  you  are  not  specifying  the
configuration using
#######  runtime switches.  Note that runtime switches do
SUPERSEED
#######  any values that you have specified here!
#######

# What path you want the .rules file containing all of the
processed
# rules? (this value has changed as of 0.4.0, previously we
copied
# all of the rules, now we are creating a single large rules
file
# but still keeping a separate file for your so_rules!
# rule_path=/usr/local/etc/snort/rules/snort.rules
rule_path=/etc/snort/rules/snort.rules

# What path you want the .rules files to be written to, this
is UNIQUE
# from the rule_path and cannot be used in conjunction, this
is to be used with the
# -k runtime flag, this can be set at runtime using the -K
flag or specified
# here.   If specified here, the -k option must also be
passed at runtime, however
# specifying -K <path> at runtime forces the -k option to
also be set
# out_path=/usr/local/etc/snort/rules/

# If you are running any rules in your local.rules file, we
need to
# know about them to properly build a sid-msg.map that will
contain your
# local.rules metadata (msg) information.  You can specify
other rules
# files that are local to your system here by adding a comma
and more paths...
# remember that the FULL path must be specified for EACH
value.
# local_rules=/path/to/these.rules,/path/to/those.rules
# local_rules=/usr/local/etc/snort/rules/local.rules
local_rules=/etc/snort/rules/local.rules

# Where should I put the sid-msg.map file?
# sid_msg=/usr/local/etc/snort/sid-msg.map
sid_msg=/etc/snort/sid-msg.map

# New for by2 and more advanced msg mapping.  Valid options
are 1 or 2
#  specify  version  2  if  you  are  running  barnyard2.2+.
Otherwise use 1
```

```
sid_msg_version=1

# Where do you want me to put the sid changelog?  This is a
changelog
#  that  pulledpork  maintains  of  all  new  sids  that  are
imported
sid_changelog=/var/log/sid_changes.log
# this value is optional

#######
#######  The below section is for so_rule processing only.
If you don't
#######  need to use them.. then comment this section out!
#######   Alternately,  if  you  are  not  using  pulledpork  to
process
#######  so_rules, you can specify -T at runtime to bypass
this altogether
#######

# What path you want the .so files to actually go to *i.e.
where is it
# defined in your snort.conf, needs a trailing slash
sorule_path=/usr/local/lib/snort_dynamicrules/

# Path  to  the  snort  binary,  we  need  this  to  generate  the
stub files
snort_path=/usr/local/bin/snort

# We need to know where your snort.conf file lives so that
we can
# generate the stub files
# config_path=/usr/local/etc/snort/snort.conf
config_path=/etc/snort/snort.conf

##### Deprecated - The stubs are now  categorically written
to the  single rule file!
# sostub_path=/usr/local/etc/snort/rules/so_rules.rules
sostub_path=/etc/snort/rules/so_rules.rules

#  Define  your  distro,  this  is  for  the  precompiled  shared
object libs!
# Valid Distro Types:
# Debian-5-0, Debian-6-0,
# Ubuntu-8.04, Ubuntu-10-4
# Centos-4-8, Centos-5-4
# FC-12, FC-14, RHEL-5-5, RHEL-6-0
# FreeBSD-7-3, FreeBSD-8-1
# OpenBSD-4-8
# Slackware-13-1
distro=RHEL-6-0

#######  This next section is optional, but probably pretty
useful to you.
```

```
#######  Please read thoroughly!

# If you are using IP Reputation and getting some public
lists, you will probably
# want to tell pulledpork where your blacklist file lives,
PP automagically will
# de-dupe any duplicate IPs from different sources.
#black_list=/usr/local/etc/snort/rules/iplists/default.black
list

# IP Reputation does NOT require a full snort HUP, it
introduces a concept whereby
# the IP list can be reloaded while snort is running through
the use of a control
# socket.  Please be sure that you built snort with the
following optins:
# -enable-shared-rep and --enable-control-socket.  Be sure
to read about how to
# configure these!  The following option tells pulledpork
where to place the version
# file for use with control socket ip list reloads!
# This should be the same path where your black_list lives!
#IPRVersion=/usr/local/etc/snort/rules/iplists

# The following option tells snort where the snort_control
tool is located.
snort_control=/usr/local/bin/snort_control

# What do you want to backup and archive?  This is a comma
separated list
# of file or directory values.  If a directory is specified,
PP will recurse
# through said directory and all subdirectories to archive
all files.
# The following example backs up all snort config files,
rules, pulledpork
# config files, and snort shared object binary rules.
backup=/etc/snort,/etc/pulledpork,/usr/local/lib/snort_dynam
icrules/

# what path and filename should we use for the backup
tarball?
# note that an epoch time value and the .tgz extension is
automatically added
# to the backup_file name on completeion i.e. the written
file is:
# pp_backup.1295886020.tgz
backup_file=/tmp/pp_backup

# Where do you want the signature docs to be copied, if this
is commented
# out then they will not be copied / extracted.  Note that
extracting them
```

```
# will add considerable runtime to pulledpork.
# docs=/path/to/base/www

# The following option, state_order, allows you to more
finely control the order
# that pulledpork performs the modify operations,
specifically the enablesid
# disablesid and dropsid functions.  An example use case
here would be to
# disable an entire category and later enable only a rule or
two out of it.
# the valid values are disable, drop, and enable.
# state_order=disable,drop,enable


# Define the path to the pid files of any running process
that you want to
# HUP after PP has completed its run.
#
pid_path=/var/run/snort.pid,/var/run/barnyard.pid,/var/run/b
arnyard2.pid
# and so on...
# pid_path=/var/run/snort_eth0.pid

# This defines the version of snort that you are using, for
use ONLY if the
# proper snort binary is not on the system that you are
fetching the rules with
# This value MUST contain all 4 minor version
# numbers. ET rules are now also dependant on this, verify
supported ET versions
# prior to simply throwing rubbish in this variable kthx!
# snort_version=2.9.0.0

# Here you can specify what rule modification files to run
automatically.
# simply uncomment and specify the apt path.
# enablesid=/usr/local/etc/snort/enablesid.conf
dropsid=/etc/pulledpork/dropsid.conf
# disablesid=/etc/pulledpork/disablesid.conf
# modifysid=/usr/local/etc/snort/modifysid.conf

# What is the base ruleset that you want to use, please
uncomment to use
# and see the README.RULESETS for a description of the
options.
# Note that setting this value will disable all ET rulesets
if you are
# Running such rulesets
# ips_policy=security

####### Remember, a number of these values are optional.. if
you don't
```

```
####### need to process so_rules, simply comment out the
so_rule section
####### you can also specify -T at runtime to process only
GID 1 rules.

version=0.7.0
```

TENTANG PENULIS

Dhani Widya Darma, lahir pada tanggal 30 Maret 1992 di Jakarta. Riwayat pendidikan yang telah ditempuh oleh penulis adalah sebagai berikut: pada tahun 1998-2004 menempuh jenjang pendidikan sekolah dasar di SD Negeri Jatimulya 11 dan pada tahun 2004-2007 menempuh jenjang sekolah menengah pertama di SMP Negeri 4 Tambun Selatan serta tahun 2007-2010 menempuh jenjang menengah kejuruan di SMK Yadika 8 Bekasi.

Semenjak masuk pada jurusan Teknik Komputer dan Jaringan pada saat menempuh pendidikan menengah kejuruan, penulis mulai tertarik pada bidang Jaringan Komputer. Perkenalan penulis dengan teman-teman pada lingkungan perkuliahan semakin membuat penulis untuk mendalami bidang *Network Engineering*. Penulis dapat dihubungi melalui e-mail dengan alamat dhaniwdarma[at]gmail.com. Terima kasih kepada para pembaca yang telah bersedia membaca karya akhir ini. Semoga bermanfaat.